ALEXANDRE KELLER ALBALUSTRO

FORMAÇÃO DE CÉLULAS EM SISTEMAS DE MANUFATURA: UMA ABORDAGEM USANDO ALGORITMOS GENÉTICOS

FLORIANÓPOLIS

2001

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

FORMAÇÃO DE CÉLULAS EM SISTEMAS DE MANUFATURA: UMA ABORDAGEM USANDO ALGORITMOS GENÉTICOS

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

ALEXANDRE KELLER ALBALUSTRO

Florianópolis, Julho de 2001.

FORMAÇÃO DE CÉLULAS EM SISTEMAS DE MANUFATURA: UMA ABORDAGEM USANDO ALGORITMOS GENÉTICOS

ALEXANDRE KELLER ALBALUSTRO

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração *em Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Marcelo Ricardo Stemmer, Dr.

Banca Examinadora:

Marcelo Ricardo Stemmer, Dr.-Ing.

Guilherme Bittencourt, Dr. Rer. Nat.

Fernando A. Ostuni Gauthier, Dr.

Para meus pais, ***Lizette e Luiz,***

que sempre foram tudo para mim...

AGRADECIMENTOS

Antes de tudo, eu gostaria de agradecer aos meus pais por tudo o que fizeram por mim. Sem eles, seria muito difícil chegar onde estou.

Agradeço aos professores do LCMI que me ajudaram a desenvolver uma forte formação acadêmica em automação e informática. Agradecimentos especiais aos professores Jean-Marie Farines, Edson R. de Pieri e José E. R. Cury. Trabalhar no LCMI superou todas as minhas expectativas.

Agradeço ao meu orientador, professor Marcelo Ricardo Stemmer, por sua paciência, confiança e liberdade para que eu pudesse realizar este trabalho.

Agradeço aos membros desta banca, pelas sugestões e críticas a este trabalho.

Agradeço a todos os meus colegas do LCMI (muitos para nomear aqui) que, de uma forma ou de outra, foram muito importantes para mim. Agradeço em especial ao Fábio Benevenuti e ao inestimável Marcos B. R. Vallim, cuja amizade foi muito importante em uma época difícil.

Eu não poderia esquecer de agradecer ao melhor amigo, Alexandre de Souza.

Agradeço ainda a todos os meus irmãos: dany, mana, rafa e fernando.

Agradeço à minha noiva Tatiane Wisintainer, pelo amor, carinho e muita paciência.

Agradeço à CAPES pela bolsa parcial concedida.

Finalmente, agradeço à DEUS por tudo. Muito obrigado.

Resumo da Dissertação apresentação à UFSC como parte dos requisitos necessários
Para a obtenção do grau de Mestre em Engenharia Elétrica.

FORMAÇÃO DE CÉLULAS EM SISTEMAS DE MANUFATURA: UMA ABORDAGEM USANDO ALGORITMOS GENÉTICOS

ALEXANDRE KELLER ALBALUSTRO

Julho/2001

Orientador: Marcelo Ricardo Stemmer, Dr.
Área de Concentração: Controle, Automação e Informática Industrial.
Palavras-Chaves: Formação de células de manufatura, algoritmo genético, têmpera simulada, heurísticas, otimização combinatória.
Número de Páginas: 159

RESUMO: Esta dissertação aborda o problema de formação de células de manufatura (PFCM). Peças que possuem características de manufatura e/ou projeto são agrupadas juntas, formando famílias de peças, e as máquinas requeridas para produzir estas peças são agrupadas dentro de células de manufatura. Nossa abordagem considera importantes aspectos da manufatura, tais como rotas alternativas de processamento, seqüência de operações, demanda de produção das peças e carga de trabalho das máquinas. Entretanto, nem todos estes aspectos são considerados juntos. Duas funções objetivos, uma para minimizar a movimentação intercelular e outra para minimizar a variação da carga de trabalho nas células são usadas para formar células de máquinas. Para resolver os modelos, nós propomos uma heurística baseada em um algoritmo genético capaz de obter soluções de boa qualidade. Os resultados obtidos em teste demonstram que a abordagem é eficiente e muito promissora.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

CELL FORMATION IN MANUFACTURING SYSTEMS: AN APPROACH USING GENETIC ALGORITHMS

ALEXANDRE KELLER ALBALUSTRO

July/2001

Advisor: Marcelo Ricardo Stemmer, Dr.

Area of Concentration: Control, Automation and Industrial Computing.

Keywords: Manufacturing cell formation, genetic algorithm, simulated annealing, heuristics, combinatorial optmization.

Number of Pages: 159

ABSTRACT: In this dissertation, the manufacturing cell formation problem is adressed. Parts which have similar processing requirements are grouped together, forming part families, and the machines required to produce these parts are grouped in machine cells. Our approach considers some important aspects of manufacturing, such as alternative routings, process sequences, workload on each machine, and production demand for each part. Two objective functions are used in this research: minimization of intercell movements and minimization of cell load variation. In order to solve these models, we propose a genetic algorithm-based heuristic which is able to find good solutions. The approach was tested on several problems from the literature and is shown to be an effective and promising tool.

SUMáRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1 COMPUTAÇÃO NA ENGENHARIA

Uma das tecnologias mais importantes desenvolvidas no século 20 foi o computador digital. Atualmente, o uso de computadores e dispositivos micro-processados é praticamente indispensável em qualquer fábrica moderna, seja de processos contínuos ou de manufatura discreta. Mais especificamente, computadores e determinados *softwares* aumentam a produtividade e qualidade de muitas tarefas da engenharia. Apenas para citar algumas:

i. Projeto e análise: Sistemas CAD permitem ao engenheiro projetista desenvolver complexos projetos, tais como moldes para injetoras, compressores, componentes automotivos, aviônicos etc., e a manipulá-los. A modelagem pode ser feita usando desde desenhos em duas dimensões até realísticos objetos sólidos em três dimensões. Através da análise de elementos finitos e simulação cinemática, pode-se analisar a performance funcional do modelo sólido 3D e verificar se ele atende aos requisitos de projeto.

ii. Avaliação de desempenho: Através do uso da simulação discreta é possível modelar e analisar o comportamento de sistemas de manufatura. É possível construir modelos de simulação bem aproximados do sistema real, capturando inclusive várias fontes de aleatoriedade do sistema, tais como tempos de intervalo entre quebras de máquinas e de reparo, permitindo ao engenheiro de processos avaliar o desempenho do sistema com respeito aos indicadores de performance previamente definidos.

iii. Otimização: A tarefa de otimização é fundamental em engenharia. O objetivo aqui é obter a melhor solução factível para um dado problema, dentro de um conjunto de soluções, que otimize um determinado critério ou objetivo. Por exemplo, no projeto pode-se minimizar o peso de um componente mecânico, enquanto que na manufatura, o objetivo pode ser maximizar o balanceamento de carga entre as máquinas do sistema. Existem diversos métodos de otimização que podem ser empregados, cuja adequação depende da aplicação: métodos baseados em derivadas, programação linear/inteira,

programação dinâmica e heurísticas diversas (têmpera simulada, algoritmos evolutivos, etc.).

iv. Armazenamento e visualização de dados: Bancos de dados permitem armazenar, manipular e visualizar dados obtidos a partir de diversas fontes dentro de uma organização. Eles constituem uma importante ferramenta de suporte para que diversas tarefas da engenharia possam ser executadas com eficiência, da área de projetos até a produção. Ferramentas de mineração de dados e OLAP (on-line analytical processing) podem ser especialmente úteis para engenheiros que freqüentemente estão envolvidos em processos de tomada de decisão estratégicas para a organização.

v. Controle: Em um sistema de manufatura flexível, as atividades das estações de processamento, tipicamente máquinas-ferramentas CNC, e do sistema de manipulação de materiais são coordenadas/controladas por computadores [1].

1.2 SISTEMAS DE MANUFATURA CELULAR

Devido ao substancial aumento da competição em praticamente todos os setores industriais, a busca por um aumento da produtividade e redução de custos tem conduzido as organizações a adotarem estratégias de manufatura e negócios mais eficientes.

A atual globalização dos mercados acentuou a necessidade das organizações assumirem uma postura mais ágil diante de cenários dinâmicos e muitas vezes críticos. Muitas destas organizações e pesquisadores têm percebido a importância de um conceito desenvolvido por S.P. Mitrofanov na antiga União Soviética para aumentar a produtividade. Este conceito é denominado de Tecnologia de Grupo (TG) e busca tomar vantagens do agrupamento de objetos com atributos similares dentro de um mesmo conjunto [2, 3]. Os atributos dos objetos podem ser características de projeto, manufatura ou ambos.

TG é uma estratégia muito simples, mas poderosa, podendo ser empregada em diversas áreas dentro de uma organização, tais como manufatura, projetos, compras e vendas [4]. Em muitas destas aplicações o uso de um sistema de codificação e classificação adequado pode ser necessário.

A engenharia de projetos, utilizando a TG, poderia empregar um sistema de codificação para descrever as peças com respeito a suas dimensões ou forma geométrica (atributos de projeto). No desenvolvimento do desenho de uma nova peça, esta codificação facilitaria a recuperação de desenhos similares em uma base de dados para serem reutilizados

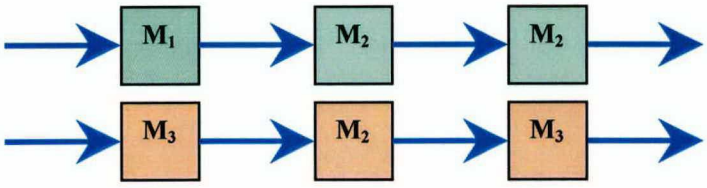
pelo projetista. A alteração de um desenho existente é muito mais econômica e rápida do que iniciar o ciclo de desenvolvimento completo de um novo desenho. Outros benefícios obtidos nesta área seriam a padronização de projetos e redução da replicação de desenhos.

A manufatura é a área de maior aplicação da TG. Manufatura Celular (MC) é o termo utilizado para nomear esta aplicação, onde famílias de peças são identificadas a partir da similaridade de atributos de projetos e/ou manufatura das peças, e as máquinas que são capazes de atender os requisitos de uma família são agrupadas dentro de células. Usualmente as células contêm máquinas que são dissimilares em suas funções e cada família de peças é atribuída a uma célula de máquina. Este problema de agrupar peças em famílias de peças e máquinas em células de máquinas é denominado de problema de formação de células de manufatura (PFCM). Toda peça que entra em uma célula deveria receber todas as operações requeridas no seu plano de processo para ser finalizada e conseqüentemente despachada para fora da célula.

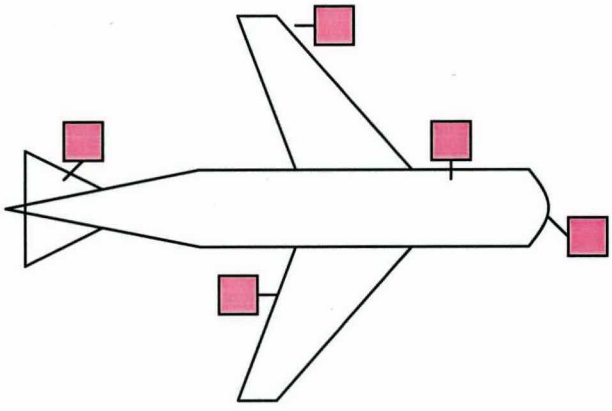
Células de manufatura que não possuem movimentos intercelulares, i.e., movimentos de peças que necessitam operações fora de suas células, são denominadas de células ideais. O transporte intercelular de peças não adiciona valor para o processo, ao contrário, aumenta o custo das peças em processo. Entretanto, é muito difícil na prática obter tal sistema de manufatura, seja ou pelo custo da aquisição de máquinas duplicadas ser proibitivo, ou pela própria limitação de espaço físico de uma fábrica.

Um sistema de manufatura que apresenta um leiaute celular é chamado de um sistema de manufatura celular. Em contraste ao tipo de arranjo utilizado por este sistema, existem outros arranjos físicos que são convencionalmente empregados, e.g. arranjo em linha, de posição fixa e funcional. O arranjo em linha caracteriza-se por dispor as máquinas da fábrica na seqüência requerida pelas peças. A indústria de produção em massa utiliza este arranjo, que possui baixa flexibilidade e alta produtividade. O arranjo de posição fixa é utilizado, por exemplo, para a produção de aviões e navios. Finalmente, em um arranjo funcional, as máquinas são agrupadas em departamentos de acordo com suas funções. Este arranjo possibilita uma alta flexibilidade e baixa produtividade, além de aumentar o custo devido à excessiva movimentação das peças entre os departamentos e ter um alto WIP (work-in-process - material em processo).

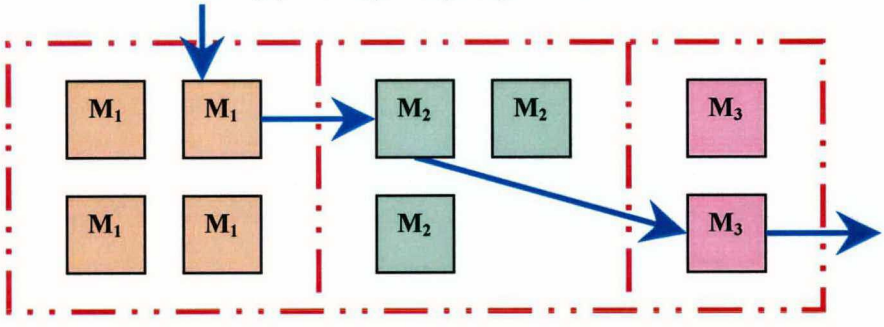
A Figura 1.1 apresenta os quatro tipos de arranjos físicos usualmente empregados nos sistemas de manufatura.



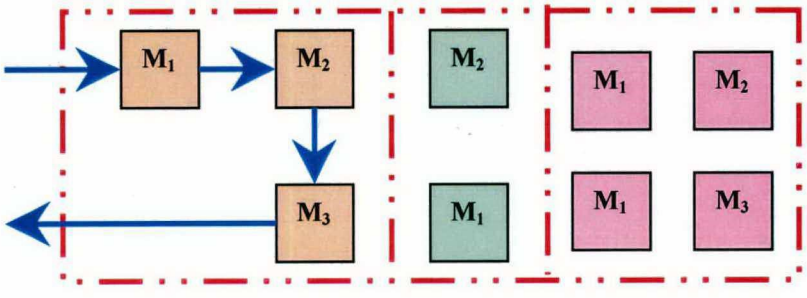
(a) Arranjo em linha.



(b) Arranjo de posição fixa.



(c) Arranjo funcional.



(d) Arranjo celular (ou em grupo).

Figura 1.1 – Tipos de arranjos mais comuns em sistemas de manufatura.

1.3 BENEFÍCIOS DA MANUFATURA CELULAR

Os benefícios obtidos com a adoção da manufatura celular foram discutidos em [5, 6, 7]. Eles podem ser resumidos da seguinte forma: menor tempo de preparo (*setup time*), menor trabalho em processo (*Work-in-process* – WIP), custos de manipulação de materiais reduzidos, menor requerimento de ferramentas por parte das peças, aumento na qualidade do produto e um melhor controle global das operações.

1.4 HEURÍSTICAS PARA RESOLVER O PFCM

O problema de agrupamento de um conjunto de máquinas (objetos) em um número pré-determinado de células é NP-completo, i. e. o PFCM é NP-completo [8, 9, 10]. Segundo a teoria da complexidade, a classe NP denota o conjunto de todos os problemas de decisão resolvíveis por um algoritmo não-determinístico em tempo polinomial [11]. Os problemas NP-completo são considerados os mais difíceis dentro da classe NP. A principal característica destes problemas é que à medida que a dimensão de entrada de um problema aumenta, o espaço de busca rapidamente torna-se imenso, o que absolutamente impossibilita uma busca exaustiva para obter a solução ótima. E, devido a isto, muitos trabalhos reportados na literatura para resolver o PFCM têm empregado heurísticas [12].

Mas afinal, o que são heurísticas? Segundo o dicionário Aurélio [13], heurística (do grego heuristiké) é:

> *1. Conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas; 2. Procedimento pedagógico pelo qual se leva o aluno a descobrir por si mesmo a verdade que lhe querem inculcar; 3. Ciência auxiliar da história que estuda a pesquisa das fontes.*

Pearl [14] apresenta a seguinte definição:

> *Heurísticas são critérios, métodos ou princípios para decidir qual, entre diversos cursos alternativos de ação, promete ser o mais efetivo a fim de atingir alguma meta. Elas representam compromissos entre dois requerimentos: a necessidade para fazer tal critério simples e, ao mesmo tempo, o desejo para ver boas e más escolhas discriminadas corretamente.*

Heurísticas não garantem a obtenção da solução ótima para o problema, mas freqüentemente fornecem boas soluções, ou até quase-ótimas, em um tempo computacional aceitável. Neste trabalho, nós discutimos o uso de diversas heurísticas para resolver o PFCM, dando especial atenção a uma técnica denominada de algoritmos genéticos.

1.5 MOTIVAÇÃO PARA A PESQUISA E OBJETIVOS

O particionamento de um sistema de manufatura em células tem sido o alvo de intensa pesquisa, principalmente porque este arranjo do sistema propicia uma boa solução de compromisso entre flexibilidade e produtividade. Ela tenta ser tão flexível quanto um sistema *job-shop* (funcional) e tão produtiva quanto um sistema de produção em massa, operando com pequenos lotes.

As universidade americanas e asiáticas há décadas pesquisam o PFCM. Entretanto, no Brasil, o uso do conceito de sistemas de manufatura celular é muito recente. A adoção por parte das indústrias aqui é gradual, mas lenta. Diversos grupos de pesquisas no Brasil pesquisam intensamente o tópico de controle em células (flexíveis) de manufatura, mas muito pouco esforço é direcionado para a fase (inicial) de projeto destes sistemas. Uma inspeção mais cuidadosa na literatura internacional permite concluir que o Brasil não possui uma participação tão significante quanto os E.U.A., Japão, Taiwan, Índia etc. na publicação de resultados científicos em relação ao PFCM. Esta carência de pesquisas sobre o projeto de sistemas de manufatura no Brasil, principalmente utilizando técnicas modernas de inteligência artificial, é a principal motivação deste trabalho, pois não parece haver dúvidas na literatura técnica de que a adoção da MC é o caminho mais promissor para a implementação de um sistema de manufatura eficiente e competitivo, mesmo que seja utilizado parcialmente, i.e., um sistema híbrido que utilize um arranjo celular e funcional ao mesmo tempo.

O objetivo fundamental desta dissertação consiste em desenvolver uma heurística baseada em algoritmos genéticos capaz de resolver o PFCM considerando importantes aspectos encontrados em um ambiente de manufatura, e.g. rotas alternativas de processos, demanda de produção dos tipos de peças, seqüência de operações e carga de trabalho das máquinas. Entretanto, nem todos estes aspectos são considerados juntos.

As características da heurística proposta incluem: atribuição dos valores de aptidão baseado no *ranking*, seleção de indivíduos por torneio, recombinação de dois pontos e multipontos, adoção de uma estratégia elitista, reinserção de indivíduos, busca local usando

têmpera simulada, tratamento de restrições com termos de penalidades e implementação no *software Matlab*© (versão 5.2). A validação da implementação consiste em resolver, de forma satisfatória, problemas obtidos da literatura.

1.6 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante desta dissertação está organizada da seguinte forma. O capítulo 2 apresenta uma revisão dos métodos utilizados para resolver o PFCM, dando uma maior ênfase para aqueles métodos que empregam inteligência computacional (IC), e.g. redes neurais não-supervisionadas, agrupamento difuso (*fuzzy clustering*) e algoritmos genéticos. Heurísticas clássicas como ROC, DCA e BEA também são apresentadas. Algoritmos e exemplos são fornecidos para auxiliar a compreensão destes métodos.

O capítulo 3 apresenta uma revisão sobre otimização combinatória, apresentando conceitos formais e heurísticas para resolver esta classe de problemas de otimização. Neste capitulo, o conceito de algoritmos evolutivos, e conseqüentemente algoritmos genéticos (AGs), são descritos sucintamente. Na seqüência, o capítulo 4 apresenta uma descrição mais detalhada sobre os AGs. O capítulo 5 apresenta uma definição formal do PFCM, a dimensão do espaço de busca para este tipo de problema combinatório e uma heurística baseada em AGs para resolver o PFCM considerando aspectos reais de um ambiente de manufatura. No capítulo 6, problemas obtidos a partir da literatura são utilizados para testar e validar a abordagem proposta nesta dissertação. Uma discussão sobre os resultados também é apresentada neste capítulo. Finalmente, o capítulo 7 apresenta as conclusões, contribuições deste trabalho e direções para trabalhos futuros nesta área.

CAPÍTULO 2

MÉTODOS PARA FORMAÇÃO DE CÉLULAS DE MANUFATURA

2.1 INTRODUÇÃO

O projeto de um sistema de manufatura celular (SMC) é uma tarefa muito complexa que envolve um estudo técnico – a definição da estrutura física das células de manufatura e o planejamento dos procedimentos operacionais – e um estudo econômico que viabilize e justifique sua implantação. Estes estudos técnico e econômico estão intimamente relacionados.

Wemmerlöv e Hyer [6] dividiram as atividades de projeto em duas grandes áreas:

1. Atividades Estruturais:
   1.1 Seleção de peças e formação de famílias de peças;
   1.2 Seleção de máquinas e processos e o agrupamento destes dentro de células;
   1.3 Seleção de ferramentas, fixações e paletes;
   1.4 Seleção dos equipamentos de manipulações de materiais
   1.5 Leiaute físico do sistema.
2. Atividades Operacionais
   2.1 Formulação de políticas de manutenção e inspeção;
   2.2 Definição dos procedimentos para planejamento da produção, escalonamento e controle;
   2.3 Definição de tarefas e atribuição das responsabilidades de tarefas para operadores;
   2.4 Definição da *interface* das células individuais com o restante do sistema de manufatura (em termos de fluxo de trabalho e informação).

Cada um destes tópicos afeta a performance e/ou custo do SMC. A formação de famílias de peças (estágio 1.1) e o agrupamento de máquinas dentro de células (estágio 1.2) é denominado problema de formação de células de manufatura (PFCM) e constitui-se como um dos maiores problemas a serem resolvidos durante o projeto do SMC.

Nas últimas décadas a comunidade científica tem pesquisado intensamente o desenvolvimento de métodos para resolver o PFCM [2, 3].

Uma classificação dos métodos existentes utilizados na resolução do PFCM, longe de ser a mais abrangente e/ou definitiva, é apresentada na Tabela 2.1.

FORMAÇÃO DE CÉLULAS DE MANUFATURA			
	ORIENTADA A PRODUÇÃO	Programação Matemática	Programação Inteira, Dinâmica etc.
		Reconhecimento de Padrões	Redes Neurais, Estatística
		Teoria de Grafos	
		Agrupamento Hierárquico	SLC, ALC, CLC, WALC etc.
		Agrupamento Não-Hierárquico	ZODIAC
		Inteligência Artificial (Simbólica)	Sistemas Especialistas
		Baseado em Arranjo	ROC/ROC2
			MODROC
			Fuzzy ROC
			DCA
			BEA
		Inteligência Computacional	Algoritmos Genéticos
			Agrupamento Difuso
			Redes Neurais Não-supervisionadas (ART1, FART e SOM)
	ORIENTADA AO PROJETO	Inspeção Visual	
		Classificação e Codificação	CODE, OPITZ, MISCLASS, etc.
			Agrupamento Difuso (IC)
			Rede Neural Direta (IC)
	ABORDAGENS	**MÉTODOS**	**FERRAMENTAS**

Tabela 2.1 – Classificação das abordagens e métodos para formação de células.

O objetivo deste capítulo é apresentar uma revisão geral dos métodos propostos na literatura para resolver o PFCM, discutindo inclusive as vantagens e desvantagens destes métodos. Algoritmos e exemplos são apresentados ao longo do texto para facilitar o entendimento do leitor.

Os métodos apresentados nesta seção podem formar células de máquinas e famílias de peças simultaneamente ou não. Neste último caso, o procedimento é seqüencial, isto é, primeiro forma-se L células de máquinas (L famílias de peças) e em seguida as L famílias de peças (L células de máquinas). Uma maior ênfase será dada aos métodos que utilizam técnicas da inteligência computacional (IC). IC é o termo sugerido por Bezdek para denominar as técnicas da inteligência artificial (IA) que utilizam uma representação numérica do conhecimento (o enfoque tradicional da IA se baseia na representação simbólica) [15].

Alguns exemplos destas técnicas são lógica difusa, redes neurais e computação evolutiva.

2.2 CLASSIFICAÇÃO E CODIFICAÇÃO

Classificação e codificação é um método para formação de famílias de peças. As peças dentro de uma fábrica são codificadas de acordo com suas características de projeto (dimensões e geometria) e/ou manufatura (necessidade de um tratamento térmico, seqüência de operações tais como *machining*, montagem, inspeção etc.) [4]. O código de uma peça é formado por uma cadeia de caracteres numéricos ou alfanuméricos que descrevem estas características. As peças que possuem o mesmo código ou similar são agrupadas juntas (classificação).

Esquemas de codificação diferem em termos dos símbolos que eles empregam e de como estes símbolos são atribuídos para gerar o código para cada peça [16]. A codificação pode ser classificada em 3 formas de estruturas:

- Monocódigo (hierárquico): Possui uma estrutura do tipo árvore, onde um dígito (nó pai) pode ter a informação ampliada por um dígito sucessor (nó filho) na ramificação;
- Policódigo (cadeia): Cada posição de um dígito representa uma informação independente dos outros dígitos;
- Multicódigo (híbrido): É uma combinação das duas primeiras estruturas.

Alguns sistemas de codificação existentes são: CODE (EUA), DCLASS (EUA), MICLASS (Holanda), Opitiz (Alemanha) e Brisch (Inglaterra) [16].

2.3 MATRIZ DE INCIDÊNCIA MÁQUINA-PEÇA

Muitos métodos desenvolvidos para resolver o PFCM, tais como ROC, DCA e BEA (discutidos na seção 2.4) operam sobre uma matriz de incidência máquina-peça binária $[a_{ij}]$. Esta matriz fornece o relacionamento entre os tipos de máquinas e os tipos de peças, representados pelas linhas e colunas de $[a_{ij}]$, respectivamente. Um elemento $a_{ij} = 1$ desta matriz indica que o tipo de peça j é processado pelo tipo de máquina i, caso contrário, o valor do elemento a_{ij} é igual a zero.

A princípio, quando uma matriz máquina-peça é construída, não existem blocos de máquinas e peças identificáveis (Figura 2.1a). É necessário a utilização de um método de agrupamento que, após realizar uma série de manipulações das linhas e colunas de $[a_{ij}]$, transforma a matriz inicial em uma (possivelmente) forma diagonal de bloco (Figura 2.1b).

Cada bloco formado ao longo da diagonal de $[a_{ij}]$ representa uma célula de máquinas (lido através das linhas) e sua respectiva família de peças (lido através das colunas), que são formadas simultaneamente.

	P_1	P_2	P_3	P_4	P_5
M_1		1		1	1
M_2	1		1		
M_3		1		1	
M_4	1		1		

(a) Matriz máquina-peça inicial: nenhum bloco diagonal identificável

		Famíla 1		Famíla 2		
		P_1	P_3	P_2	P_4	P_5
Célula 1	M_2	1	1			
	M_4	1	1			
Célula 2	M_1			1	1	1
	M_3			1	1	

(b) Matriz máquina-peça rearranjada em uma (perfeita) forma diagonal de blocos.

Figura 2.1 – Identificação de células de manufatura e famílias de peças a partir da matriz máquina-peça.

A parir da Figura 2.1b é possível distinguir duas células, célula 1 e célula 2, e as duas famílias de peças correspondentes a estas células, família 1 e família 2. É importante ressaltar que os blocos formados na Figura 2.1b são mutuamente separáveis, isto é, não há tipos de peças da célula 1 que necessitam visitar máquinas da célula 2 e vice-versa. Neste caso especial não existem movimentos intercelulares, entretanto, isto é raro em ambientes de manufatura reais.

A Figura 2.2a apresenta uma matriz $[a_{ij}]$ que não pode ser decomposta em um número de blocos diagonais separáveis devido a presença de elementos excepcionais (tipos de peças que requerem processamento em duas ou mais células). Os tipos de máquinas que atendem os tipos de peças excepcionais são denominados de máquinas gargalos, uma vez que mais de uma família de peças compartilham este recurso. Alguns autores sugeriram que os tipos de peças excepcionais possam ser subtraídas do problema ou que os tipos de máquinas gargalos sejam replicadas [11]. Este último caso é ilustrado na Figura 2.2b.

		Famíla 1		Famíla 2		
		P_1	P_3	P_2	P_4	P_5
Célula 1	M_2	1	1			1
	M_4	1	1			
Célula 2	M_1			1	1	1
	M_3			1	1	

(a) Matriz máquina-peça com um elemento excepcional ($a_{25} = 1$).

		Famíla 1		Famíla 2		
		P_1	P_3	P_2	P_4	P_5
Célula 1	M_2	1	1			
	M_4	1	1			
Célula 2	M_1			1	1	1
	M_3			1	1	
	M_2					1

(b) Matriz máquina-peça com 1 tipo de máquina duplicada (M_2) e atribuída à célula 2.

Figura 2.2 – Duplicação de máquinas para eliminar movimentos intercelulares.

Os elementos binários da matriz máquina-peça podem ser substituídos por outros tipos de informações relevantes ao PFCM, tais como:

- tempo de processamento do tipo de peça j no tipo de máquina i ($t_{ij} > 0$);
- carga de trabalho induzida pela demanda de produção do tipo de peça j no tipo de máquina i ($w_{ij} > 0$);

- seqüência de tipos de máquinas visitadas pelo tipo de peça j.

Venugopal e Narendran [8, 9] definem a carga de trabalho w_{ij} (admensional) como sendo

$$w_{ij} = {(t_{ij} * N_j)}/{T_i}, \text{ onde}$$

t_{ij} = Tempo de processamento do tipo de peça j no tipo de máquina i (hora/unidade);
N_j = Demanda de produção do tipo de peça j para um dado período de tempo (unidade/ano);
T_i = Tempo disponível do tipo de máquina i para um dado período de tempo (hora/ano);

$$a_{ij} = \begin{cases} w_{ij} > 0; \\ 0, \text{ caso contrário.} \end{cases}, \text{ é a matriz de carga de trabalho.}$$

Heragu [3] define k (inteiro positivo) como sendo o número da operação para o qual o tipo de peça j visita o tipo de máquina i:

$$a_{ij} = \begin{cases} k, \text{ se i é visitado por j para realizar a k - ésima operação;} \\ 0, \text{ caso contrário.} \end{cases}$$

Entretanto, estas duas definições somente são válidas quando um tipo de máquina i não é visitado pelo tipo de peça j para realizar operações não-consecutivas (operações consecutivas no mesmo tipo de máquina podem ser tratadas como sendo uma única operação, e, portanto, são permitidas).

2.4 MÉTODOS BASEADOS EM ARRANJO

Métodos baseados em arranjos buscam identificar células de manufatura e família de peças simultaneamente através da manipulação de linhas e colunas da matriz máquina-peça. Os métodos mais importantes desta classe são: ROC *(Rank Order Clustering)*, DCA *(Direct Clustering Algorithm)* e BEA *(Bond Energy Analysis)*. Chu e Tsai [17] apresentaram um estudo comparativo do uso destes métodos e Cheng [18] reportou um estudo adicional, o qual incluiu outros métodos como SLC *(Single Linkage Clustering)* e ZODIAC.

2.4.1 *RANK ORDER CLUSTERING* (ROC)

King [19] desenvolveu um algoritmo onde linhas e colunas de uma matriz binária máquina-peça são lidas como palavras binárias e recebem os seus valores decimais correspondentes. Durante um número finito de iterações, as linhas e colunas são rearranjadas segundo estes valores decimais para no final estarem dispostas em ordem crescente e formarem blocos diagonais, caso eles existam, na matriz máquina-peça. O algoritmo ROC é dado a seguir [20]:

Algoritmo ROC

Entrada: $[a_{ij}]_{m*n}$ ***(Matriz inicial)***

Saída: $[a_{ij}]_{m*n}$ ***(Matriz Final)***

Passo 1: Calcule o peso total da coluna W_j *(j=1:n):*

$$W_j = \sum_{i=1}^{m} 2^i \cdot a_{ij}$$

Passo 2: Se (W_j *não está em ordem crescente) então*

{Ordene as colunas de $[a_{ij}]$ *para fazer* W_j *em ordem crescente}*

Passo 3: Calcule o peso total da linha W_i *(i=1:m):*

$$W_i = \sum_{j=1}^{n} 2^j \cdot a_{ij}$$

Passo 4: Se (W_i *não está em ordem crescente) então*

{Ordene as linhas de $[a_{ij}]$ *para fazer* W_i *em ordem crescente}*

Passo 5: Se ($[a_{ij}]$ *está em uma forma diagonal de blocos) então pare*

Senão retorne ao passo 1.

Exemplo 1: Considere a matriz máquina-peça inicial da figura 2.3(a). O uso do algoritmo ROC para encontrar blocos diagonais nesta matriz é apresentado a seguir.

	P_1	P_2	P_3	P_4	P_5	P_6
M_1		1		1		1
M_2					1	1
M_3		1			1	1
M_4	1		1			
M_5	1		1			
M_6		1			1	1
O	2	4	3	1	5	6

(a) Matriz inicial.

(j =1) $W_1 = 2^4 + 2^5 = 48$

(j =2) $W_2 = 2^1 + 2^3 + 2^6 = 74$

(j =3) $W_3 = 2^4 + 2^5 = 48$

(j =4) $W_4 = 2^1 = 2$

(j =5) $W_5 = 2^2 + 2^3 + 2^6 = 76$

(j =6) $W_6 = 2^1 + 2^2 + 2^3 + 2^5 = 78$

(b) Passo 1/1º Iteração.

	P_4	P_1	P_3	P_2	P_5	P_6	O
M_1	1			1		1	3
M_2					1	1	4
M_3				1	1	1	5
M_4		1	1				1
M_5		1	1				2
M_6				1	1	1	6

(c) Ordenamento da matriz (Passo 2/1º Iteração).

(i =1) $W_1 = 2^1 + 2^4 + 2^6 = 82$

(i =2) $W_2 = 2^5 + 2^6 = 96$

(i =3) $W_3 = 2^4 + 2^5 + 2^6 = 112$

(i =4) $W_4 = 2^2 + 2^3 = 12$

(i =5) $W_5 = 2^2 + 2^3 = 12$

(i =6) $W_6 = 2^4 + 2^5 + 2^6 = 112$

(d) Passo 3/1º Iteração.

	P_4	P_1	P_3	P_2	P_5	P_6
M_4		1	1			
M_5		1	1			
M_1	1			1		1
M_2					1	1
M_3				1	1	1
M_6				1	1	1
O	3	1	2	4	5	6

(e) Ordenamento da matriz (Passo 4/1º Iteração).

(j =1) $W_1 = 2^3 = 8$

(j =2) $W_2 = 2^1 + 2^2 = 6$

(j =3) $W_3 = 2^1 + 2^2 = 6$

(j =4) $W_4 = 2^3 + 2^5 + 2^6 = 104$

(j =5) $W_5 = 2^4 + 2^5 + 2^6 = 112$

(j =6) $W_6 = 2^3 + 2^4 + 2^5 + 2^6 = 120$

(f) Passo 1 (2º Iteração).

Figura 2.3 – Exemplo usando o algoritmo ROC.

	P_1	P_3	P_4	P_2	P_5	P_6	O
M_4	1	1					1
M_5	1	1					2
M_1			1	1		1	3
M_2					1	1	4
M_3				1	1	1	5
M_6				1	1	1	6

(g) Ordenamento da matriz (Passo 2/2º Iteração).

(i =1) $W_1 = 2^1 + 2^2 = 6$

(i =2) $W_2 = 2^1 + 2^2 = 6$

(i =3) $W_3 = 2^3 + 2^4 + 2^6 = 88$

(i =4) $W_4 = 2^5 + 2^6 = 96$

(i =5) $W_5 = 2^4 + 2^5 + 2^6 = 112$

(i =6) $W_6 = 2^4 + 2^5 + 2^6 = 112$

(4) Passo 3/2º Iteração.

Figura 2.3 – Exemplo usando o algoritmo ROC (continuação).

Conforme a figura 2.3(g), a matriz $[a_{ij}]$ já possui uma forma diagonal de blocos, então o processo iterativo do algoritmo ROC é finalizado.

Como pode ser visto na figura 2.3(g), as máquinas M_4 e M_5 formam uma célula de manufatura (P_1 e P_3 formam a correspondente família de peças), enquanto que as máquinas M_1, M_2, M_3 e M_6 formam a outra célula (P_2, P_5, P_4 e P_6 formam a correspondente família de peças).

Segundo King e Nakornchai [21], algoritmo ROC tem uma complexidade computacional de ordem cúbica, mais especificamente, $O(m^2n+n^2m)$ (onde m e n representam os números das linhas e colunas, respectivamente). Este método apresenta diversas limitações, tais como:

⇒ Não considera nenhum importante dado de manufatura como demanda de produção, tempo de processamento, seqüência de operações, tamanho da célula etc. (outros métodos baseados em arranjo também não consideram) [2];

⇒ A presença de elementos excepcionais, e correspondentemente máquinas-gargalos, prejudica a manipulação das linhas e colunas da matriz máquina-peça. Portanto, eles devem inicialmente ser identificados e temporariamente ignorados de maneira que o método possa continuar a ser aplicado à matriz [21]. Após a obtenção de uma matriz final, os elementos excepcionais anteriormente ignorados são marcados na matriz por asteriscos em vez de uns;

⇒ A qualidade do resultado final é dependente da matriz máquina-peça [11];

⇒ Bons resultados somente podem ser produzidos se os elementos excepcionais forem corretamente identificados, o que pode muitas vezes requerer a intervenção humana (o mesmo vale para DCA e BEA) [18];

⇒ Uma desvantagem intrínseca ao algoritmo ROC é a necessidade de armazenar as palavras binárias e ter que realizar o ordenamento das linhas e colunas.

2.4.2 *DIRECT CLUSTERING ALGORITHM* (DCA)

Chan e Milner [22] introduziram o DCA, um algoritmo que posiciona as linhas com elementos positivos ($a_{ij} = 1$) mais à esquerda para o topo, enquanto que as colunas com os elementos positivos mais a cima são posicionados para a esquerda da matriz. Resultados idênticos são obtidos a partir de qualquer forma da matriz máquina-peça inicial, ao contrário do algoritmo ROC [11]. O algoritmo DCA é dado a seguir [20]:

Algoritmo DCA

Entrada: $[a_{ij}]_{m*n}$ ***(Matriz inicial)***

Saída: $[a_{ij}]_{m*n}$ ***(Matriz Final)***

Passo 1: Para $\forall i$, calcule o número total de elementos não-nulos na linha i:

$$W_i = \sum_{j=1}^{n} a_{ij}$$

Passo 1.1: Ordene as linhas em ordem descendente.

Passo 2: Para $\forall j$, calcule o número total de elementos não-nulos na coluna j:

$$W_j = \sum_{i=1}^{m} a_{ij}$$

Passo 2.1: Ordene as colunas em ordem ascendente.

Passo 3: Para i=1:m, mova todas as colunas, onde $a_{ij} = 1$, para a direita, mantendo a ordem das linhas anteriores.

Passo 4: Para j=1:n, mova todas as linhas, onde $a_{ij} = 1$, para o topo, mantendo a ordem das colunas anteriores.

Passo 5: Se {a matriz corrente é a mesma que a anterior} então pare,
Senão retorne ao passo 3.

Exemplo 2: Considere a matriz máquina-peça inicial da figura 2.3(a) (a mesma utilizada no exemplo 1, seção ROC). O uso do algoritmo DCA para encontrar blocos diagonais nesta matriz é apresentado a seguir.

	P_1	P_2	P_3	P_4	P_5	P_6
M_1		1		1		1
M_2					1	1
M_3		1			1	1
M_4	1		1			
M_5	1		1			
M_6		1			1	1

(a) Matriz inicial.

	P_1	P_2	P_3	P_4	P_5	P_6	Σl
M_1		1		1		1	3
M_2					1	1	2
M_3		1			1	1	3
M_4	1		1				2
M_5	1		1				2
M_6		1			1	1	3

(b) Passo 1.

	P_1	P_2	P_3	P_4	P_5	P_6	Σl
M_1		1		1		1	3
M_3		1			1	1	3
M_6		1			1	1	3
M_2					1	1	2
M_4	1		1				2
M_5	1		1				2
ΣC	2	3	2	1	3	4	

(c) Passo 1.1/2.

	P_4	P_1	P_3	P_2	P_5	P_6	Σl
M_1	1			1		1	3
M_3				1	1	1	3
M_6				1	1	1	3
M_2					1	1	2
M_4		1	1				2
M_5		1	1				2
ΣC	1	2	2	3	3	4	

(d) Passo 2.1.

	P_1	P_3	P_4	P_2	P_5	P_6
M_1			1	1		1
M_3				1	1	1
M_6				1	1	1
M_2					1	1
M_4	1	1				
M_5	1	1				

(e) Matriz máquina-peça final.

Figura 2.4 – Exemplo usando o algoritmo DCA.

2.4.3 *BOND ENERGY ANALYSIS* (BEA)

McCormick et al. [23] desenvolveram um algoritmo de agrupamento de propósito geral que pode ser aplicado em qualquer matriz de números não-negativos. O Método BEA busca identificar e visualizar agrupamentos naturais que ocorrem em arranjos de dados complexos. Uma função que quantifica a densidade dos elementos para um determinado arranjo, denominada medida de efetividade (ME), é dada por:

$$ME = \left(\frac{1}{2}\right) \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \cdot \left[a_{i(j-1)} + a_{i(j+1)} + a_{(i-1)j} + a_{(i+1)j}\right]$$

onde

m = número de linhas,

n = número de colunas e

$a_{0j} = a_{(m+1)j} = a_{i0} = a_{i(n+1)} = 0$ (por convenção).

Um arranjo que possui agrupamentos densos de elementos numericamente maiores tem um valor de ME mais alto do que um arranjo no qual a disposição destes elementos está mais dispersa. ME explora a ligação entre um elemento no arranjo e seus vizinhos imediatos.

A maximização da função ME através das permutações de linhas e colunas permite que os elementos maiores no arranjo sejam agrupados juntos (energias de ligação mais fortes). Desta forma, o método BEA busca maximizar a função ME sob todos os m!n! arranjos possíveis que podem ser obtidos a partir das permutações de linhas e colunas de um arranjo inicial. Este problema pode ser decomposto em dois problemas separados de otimização, um para as linhas e outro para as colunas, pois segundo McCormick et al. [23], as ligações verticais (horizontais) não são afetadas pelo rearranjo das colunas (linhas).

Lenstra [24] mostrou que o agrupamento de um arranjo m*n não negativo obtido pela permutação de suas linhas e colunas é equivalente a dois problemas do caixeiro viajante (um para a maximização das linhas e o outro para as colunas). Portanto, este problema é NP-completo. O algoritmo BEA é apresentado a seguir [23].

Algoritmo BEA

Entrada: $[a_{ij}]_{m*n}$ ***(Matriz inicial)***

Saída: $[a_{ij}]_{m*n}$ ***(Matriz Final)***

Passo 1: Selecionar uma das colunas arbitrariamente. Ajustar $j = 1$ *e* $i = 1$*;*

Passo 2: Tentar colocar individualmente cada uma das colunas N-j restantes em cada uma das possíveis j+1 posições (para à esquerda e direita das i colunas já colocadas).

Computar a contribuição de cada coluna para a ME e posicionar a coluna que fornece o maior valor incremental para ME ao lado da coluna i;

Passo 3: Se $j = N$ *então vá para o passo 4, senão incremente j e retorne ao passo 2;*

Passo 4: Repetir o passo 2 considerando as linhas em vez das colunas;

Passo 5: Se $i = M$ *então pare, senão incremente i e retorne ao passo 4.*

2.5 MÉTODOS BASEADOS EM AGRUPAMENTOS HIERÁRQUICOS E NÃO-HIERÁQUICOS

Estes métodos usam coeficientes de similaridades para definir os relacionamentos entre objetos (máquinas, peças, ferramentas etc.) para, baseados nestas medidas de similaridade, formar família de peças e células de manufatura [2]. Referências para os métodos de agrupamentos não-hieráquicos podem ser encontrados em [11].

Métodos baseados em agrupamentos hierárquicos não formam células de máquinas e famílias de peças simultaneamente, mais especificamente, eles envolvem dois estágios: o primeiro calcula os coeficientes de similaridade entre cada par de objetos e o segundo consiste em utilizar um método de agrupamento para agrupar o par de objetos com medidas de similaridade mais altas dentro de uma partição. A partição inteira é considerada como se ela fosse um único objeto. Um nível de similaridade deve ser especificado pelo usuário, o qual permite definir o número de partições formadas.

O tipo de algoritmo de agrupamento mais importante e utilizado é denominado de aglomerativo. Eles iniciam com partições unitárias e procedem juntando-as em partições maiores até que uma partição contendo o conjunto inteiro é obtida ou que os coeficientes de similaridade fiquem abaixo do nível de similaridade especificado pelo usuário [11]. Os principais algoritmos de agrupamento são [25]: SLC (*Single Linkage Clustering*); ALC (*Average Linkage Clustering*); CLC (*Complete Linkege Clustering*) e WALC (*Weighted Average Linkage Clustering*).

2.6 REDES NEURAIS ARTIFICIAIS

2.6.1 TEORIA DE RESSONÂNCIA ADAPTATIVA (*ADAPTIVE RESONANCE THEORY 1 - ART1*)

Carpenter e Grossberg [26, 27] desenvolveram modelos de redes neurais não-supervionadas que, baseadas na teoria de ressonância adaptativa introduzida por Grossberg, resolvem o dilema estabilidade-plasticidade comumente encontrado no projeto de sistemas de aprendizado artificiais, especificamente redes neurais. Este dilema refere-se à busca de um compromisso entre o aprendizado contínuo de novos padrões relevantes (modo plástico) e a preservação do conhecimento adquirido anteriormente frente à exposição de padrões irrelevantes ou freqüentemente repetidos (modo estável). As redes do tipo ART são arquiteturas capazes de auto-organizar códigos de reconhecimento estáveis em resposta a uma arbitrária seqüência de padrões de entrada [26, 27]. Estes tipos de redes são estáveis e plásticas. A tabela 2.2 apresenta os principais modelos de redes neurais da família ART.

Redes neurais diretas (*feedforward*) não são estáveis em um ambiente dinâmico, pois os pesos da rede são ajustados e fixados durante a fase de treinamento para um determinado conjunto de padrões. Quando ocorre uma mudança no ambiente (e consequentemente nos padrões de entrada), há uma rápida degradação na performance da rede direta, sendo necessário um novo ajuste nos pesos da rede para habilitá-la a reconhecer os novos padrões. Portanto, este novo treinamento implica a perda de toda a informação previamente armazenada [27].

A figura 2.5 ilustra a arquitetura do modelo ART1. A camada F_1 é constituída de duas subcamadas, denominadas $F_{1(a)}$ (subcamada de entrada) e $F_{1(b)}$ (subcamada de comparação), onde cada nodo neural de $F_{1(a)}$ é conectado ao nodo neural correspondente em $F_{1(b)}$. Os nodos neurais de $F_{1(b)}$ e F_2 são totalmente conectados através de dois tipos de conexões ponderadas: ascendentes (*bottom-up*, $F_{1(b)} \rightarrow F_2$) e descendentes (*top-down*, $F_{1(b)} \leftarrow F_2$). Estes pesos são sempre atualizados quando a rede aprende (i.e. categoriza) um padrão de entrada X. A descrição da arquitetura básica e operação da rede neural ART1 apresentada aqui é baseada no texto de Carpenter e Grossberg [26, 27] e Fausett [28].

Tabela 2.2 – Modelos de redes neurais do tipo ART.

Modelos	Função	Referências
ART1	Categorizar padrões binários.	[26, 27, 28, 29]
ART2	Categorizar padrões binários e discretos.	[27, 28, 29, 30]
ART3	Realizar busca paralela de reconhecimento distribuídos em uma rede hierárquica multinível.	[32]
FUZZY ART	Categorizar padrões binários e discretos.	[31]
ARTMAP	Categorizar padrões com supervisão.	[33]
FUZZY ARTMAP	Categorizar padrões com supervisão.	[34]

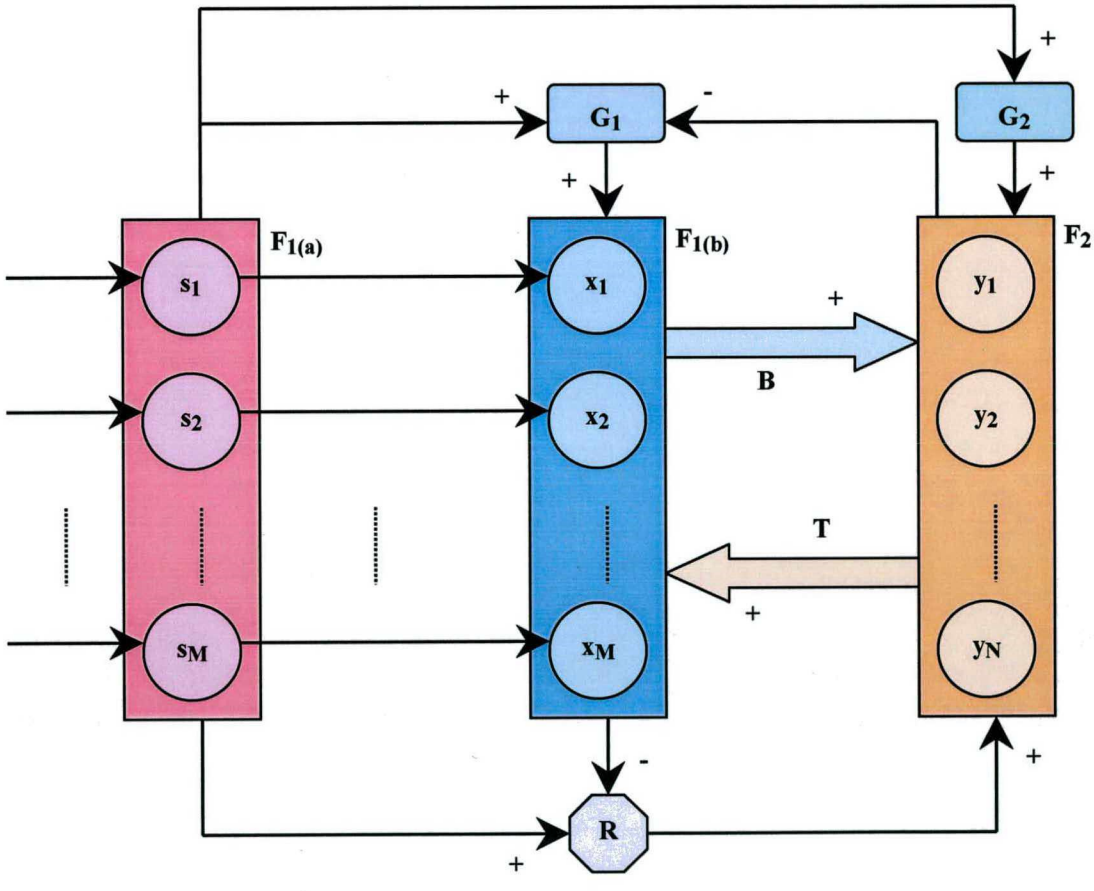


Figura 2.5 – Arquitetura da rede neural ART1. Adaptado de Carpenter e Grossberg [27] e Fausett [29].

Notação:

$F_{1(a)}$ = Subcamada de F_1 que contém os nodos neurais de entrada.

$F_{1(b)}$ = Subcamada de F_1 que contém os nodos neurais de comparação.

F_2 = Camada que contém os nodos neurais competidores.

M = número de nodos da camada F_1 (igual para cada subcamada, $F_{1(a)}$ e $F_{1(b)}$).

N = Número de nodos da camada F_2 (representa o número máximo possível de agrupamentos que a rede pode formar).

T = Vetor de N*M componentes representando os pesos de cada conexão a partir do j-ésimo nodo de F_2 para o i-ésimo nodo de $F_{1(b)}$.

B = Vetor de M*N componentes representando os pesos de cada conexão a partir do i-ésimo nodo de $F_{1(b)}$ para o j-ésimo nodo de F_2.

I = Vetor do padrão de entrada binário apresentado à rede.

G_1 = Controle de ganho da subcamada $F_{1(b)}$.

G_1 = Controle de ganho da camada F_2.

R = Mecanismo de ajuste (reset) da camada F_2.

Antes de detalharmos o funcionamento da ART1, é importante ressaltar que estamos considerando que esta rede neural está sendo operada no seu modo de aprendizado rápido. Isto é, os pesos da ART1 atingem o equilíbrio durante a apresentação de cada vetor de entrada e, desta forma, não é preciso resolver um sistema de equações diferenciais não-lineares. O processo de aprendizado se inicia com a apresentação de um padrão de entrada X à subcamada $F_{1(a)}$ (passo 2 do algoritmo ART1), cujos nodos ativos enviam dois sinais excitatórios para G_1 e R, e enviam também um padrão de atividade X para $F_{1(b)}$. Todos os sinais excitatórios enviados para R são somados, entretanto, o sistema é projetado de maneira que os nodos ativados em $F_{1(b)}$ enviam um sinal inibitório que combinados é maior e, desta forma, previne R de ser disparado [27]. Os nodos ativos de $F_{1(b)}$ recebem um sinal excitatório de G_1 e enviam um padrão de atividade ascendente X para F_2 (X é transformado em Y = B*X pelo vetor de pesos ascendentes B, passo 4 do algoritmo ART1). Aqui é interessante salientar que cada nodo de $F_{1(b)}$ possui 3 fontes das quais eles podem receber um sinal (Figura 3.3): de um nodo de $F_{1(a)}$ (sinal ascendente), de um nodo de F_2 (sinal descendente) ou do controle de ganho G_1. Similarmente, para cada nodo de F_2 (Figura 3.3): de um nodo $F_{1(b)}$ (sinal ascendente), do controle de ganho G_2 ou do mecanismo de ajuste R.

Os nodos de $F_{1(b)}$ podem enviar sinais para F_2 (e vice-versa) somente se ao menos dois sinais excitatórios forem recebidos das três fontes possíveis. Isto é denominado de regra 2/3 [26, 27]. A camada F_2 é uma camada competitiva, isto é, os nodos competem entre si para se tornarem ativos. O nodo que receber a maior ativação (passo 5 do algoritmo ART1) é escolhido o vencedor (J) e torna-se um candidato para aprender o padrão de entrada X, enquanto que os outros nodos permanecem inativos. Um padrão de atividade descendente é enviado de F_2 para F_1 ($Z = T*X$), a qual inibe G_1.

O nodo J irá efetivamente aprender X somente se seu vetor peso descendente T for similar a X (passo 6 do algoritmo ART1), de acordo com um limite de vigilância (ρ) especificado pelo usuário. Este limite determina o grau de similaridade esperado entre T e X (ou o grau de dissimilaridade que é tolerado).

Os nodos em $F_{1(b)}$ que permanecem ativos são aqueles cuja a intersecção de bits dos padrões de atividade X e Z não é zero. Se a razão $||Z||/||X||$ é maior que ρ (passo 6 do algoritmo ART1), então os padrões T e X são considerados similares e o nodo J tem seus pesos atualizados (passo 7 do algoritmo ART1). Caso contrário, $F_{1(b)}$ não é mais capaz de inibir R e isto permite que ele dispare. A função de R é inibir o nodo J que falhou no teste de similaridade e permitir que outro nodo possa ser um candidato para aprender o padrão X. Quando o nodo rejeitado J é inibido, o padrão de atividade Z e a inibição de G_1 são ambos eliminados. O padrão de atividade X é novamente enviado de $F_{1(a)}$ para $F_{1(b)}$ e um novo ciclo de teste é repetido (da mesma forma como descrito anteriormente, com exceção que agora nenhum nodo inibido participa do processo de competição).

O ciclo de teste de hipótese somente é finalizado quando: (1) um nodo de F_2 for similar a X, (2) um nodo de F_2 ainda não utilizado é selecionado ou (3) toda a capacidade de memória da ART1 (nodos de F_2) foi usada, não sendo possível acomodar o padrão X sob nenhum dos nodos de F_2 [27].

Toda vez que um vetor T atualiza seus vetores pesos B e T, os nodos de F_2 inibidos são habilitados para competirem na apresentação de um novo padrão de entrada X (passo 8 do algoritmo ART1). O algoritmo ART1 é apresentado a seguir [35].

Algoritmo ART1

Entrada: $[a_{ij}]_{m*n}$ ***(Matriz inicial)***

Saída:

Passo 1: Inicializar os pesos das conexões descendentes (t_{ji}) *e ascendentes* (b_{ij})*; selecionar o limite de vigilância* ρ*:*

$\tau = 1,\ t_{ji}(\tau) = 1,\ b_{ij}(\tau) = 1/(1+N);$

$0 \leq \rho \leq 1;$

Passo 2: Aplicar um padrão de entrada binário X à rede neural ART1:

Passo 3: Ajustar as ativações de todos os nodos em F_2 *para zero (j=1:N):*

$$y_j = 0;$$

Passo 4: Para cada nodo $j \in F_2$ *que não está inibido* ($y_j \neq -1$)*:*

$$y_j = \sum_{i=1}^{M} b_{ij}(\tau) \cdot x_i;$$

Passo 5: Selecionar o nodo j vencedor (y_j^*)*:*

$$y_j^* = \max_j \{y_j\},\ J = j;$$

Passo 6: Teste de Vigilância:

$\|X\| = \sum_{i=1}^{M} x_i$ *(número de elementos não-nulos de X);*

$\|T \cdot X\| = \sum_{i=1}^{M} t_{Ji}(\tau) \cdot x_i$ *(número de elementos não-nulos em posições em comum no vetor T e X);*

Se $\left\{ \left[\|T \cdot X\| / \|X\| \right] \leq \rho \right\}$ *então* $\{y_J = -1\}$ *(inibir o nodo J} e retorne para o passo 4, senão continue;*

Passo 7: Atualizar os pesos para o nodo J (aprendizado rápido):

(7.1) $t_{Ji}(\tau + 1) = t_{Ji}(\tau) \cdot x_i,$

(7.2) $b_{iJ}(\tau + 1) = \left[t_{Ji}(\tau) \cdot x_i / (0.5 + \|T \cdot X\|) \right];$

Passo 8: Se {ainda existe padrões de entrada} então {habilite qualquer nodo j desabilitado no passo 7 e retorne ao passo 2}, senão pare.

Exemplo 3: Considere a matriz de incidência máquina-peça mostrada na figura 2.6. O uso do algoritmo ART1 para encontrar células de máquinas nesta matriz é apresentado a seguir.

	P_1	P_2	P_3	P_4
M_1	1		1	
M_2		1		1
M_3	1			

Figura 2.6 – Matriz de incidência máquina-peça.

1º iteração ($\tau = 1$).

Passo 1:

N=3; M = 4; $\rho = 0.7$; (i=1:M; j=1:N): $t_{ji}(\tau) = 1$; $b_{ij}(\tau) = 1/(1+3) = 0.25$;

1	1	1	1
1	1	1	1
1	1	1	1

(a) Descendentes.

0.25	0.25	0.25
0.25	0.25	0.25
0.25	0.25	0.25
0.25	0.25	0.25

(b) Ascendentes.

Figura 2.7 – Matrizes de pesos inicial.

Passo 2:

$X_1 = [1\ 0\ 1\ 0]$;

Passo 3:

$y_j = 0$ (j=1:N);

Passo 4:

$y_1 = b_{11}(\tau)*x_{11} + b_{21}(\tau)*x_{12} + b_{31}(\tau)*x_{13} + b_{41}(\tau)*x_{14} = 0.5$;

$y_2 = b_{12}(\tau)*x_{11} + b_{22}(\tau)*x_{12} + b_{32}(\tau)*x_{13} + b_{42}(\tau)*x_{14} = 0.5$;

$y_3 = b_{13}(\tau)*x_{11} + b_{23}(\tau)*x_{12} + b_{33}(\tau)*x_{13} + b_{43}(\tau)*x_{14} = 0.5$;

Passo 5:

No caso de empate escolher o nodo de menor índice (critério de desempate): J = 1;

Passo 6:

$||X_1||$; $||T_1*X_1|| = 2$; $||T_1*X_1|| / ||X_1|| = 2/2 = 1 > 0.7$;

X_1 (M_1) é categorizada sob o 1º nodo.

Passo7:

7.1: Atualização dos pesos descendentes (T). Ver Figura 2.8(a).

$t_{11}(\tau+1) = t_{11}(\tau)*x_{11} = 1;$

$t_{12}(\tau+1) = t_{12}(\tau)*x_{12} = 0;$

$t_{13}(\tau+1) = t_{13}(\tau)*x_{13} = 1;$

$t_{14}(\tau+1) = t_{14}(\tau)*x_{14} = 0;$

7.2: Atualização dos pesos ascendentes (B). Ver Figura 2.8(b).

$b_{11}(\tau+1) = t_{11}(\tau)*x_{11} / (0.5 + \|T_1*X_1\|) = 0.4;$

$b_{21}(\tau+1) = t_{12}(\tau)*x_{12} / (0.5 + \|T_1*X_1\|) = 0;$

$b_{31}(\tau+1) = t_{13}(\tau)*x_{13} / (0.5 + \|T_1*X_1\|) = 0.4;$

$b_{41}(\tau+1) = t_{14}(\tau)*x_{14} / (0.5 + \|T_1*X_1\|) = 0;$

Passo 8: Ainda existem padrões a serem apresentados. Voltar ao passo 2;

1	0	1	0
1	1	1	1
1	1	1	1

(a) Descendentes.

0.4	0.25	0.25
0	0.25	0.25
0.4	0.25	0.25
0	0.25	0.25

(b) Ascendentes.

Figura 2.8 – Matrizes de pesos da 1º iteração.

2º iteração ($\tau = 2$).

Passo 2:

$X_2 = [0\ 1\ 0\ 1];$

Passo 3:

$y_j = 0$ (j=1:N);

Passo 4:

$y_1 = b_{11}(\tau)*x_{21} + b_{21}(\tau)*x_{22} + b_{31}(\tau)*x_{23} + b_{41}(\tau)*x_{24} = 0;$

$y_2 = b_{12}(\tau)*x_{21} + b_{22}(\tau)*x_{22} + b_{32}(\tau)*x_{23} + b_{42}(\tau)*x_{24} = 0.5;$

$y_3 = b_{13}(\tau)*x_{21} + b_{23}(\tau)*x_{22} + b_{33}(\tau)*x_{23} + b_{43}(\tau)*x_{24} = 0.5;$

Passo 5:

No caso de empate escolher o nodo de menor índice (critério de desempate): J = 2;

Passo 6:

$\|X_1\|$; $\|T_1 * X_1\| = 2$; $\|T_1 * X_1\| / \|X_1\| = 2/2 = 1 > 0.7$;

X_2 (M_2) é categorizada sob o 2º nodo.

Passo7:

7.1: Atualização dos pesos descendentes (T). Ver Figura 2.9(a).

$t_{11}(\tau+1) = t_{11}(\tau) * x_{21} = 0$;

$t_{12}(\tau+1) = t_{12}(\tau) * x_{22} = 1$;

$t_{13}(\tau+1) = t_{13}(\tau) * x_{23} = 0$;

$t_{14}(\tau+1) = t_{14}(\tau) * x_{24} = 1$;

7.2: Atualização dos pesos ascendentes (B). Ver Figura 2.9(b).

$b_{11}(\tau+1) = t_{11}(\tau) * x_{21} / (0.5 + \|T_1 * X_1\|) = 0$;

$b_{21}(\tau+1) = t_{12}(\tau) * x_{22} / (0.5 + \|T_1 * X_1\|) = 0.4$;

$b_{31}(\tau+1) = t_{13}(\tau) * x_{23} / (0.5 + \|T_1 * X_1\|) = 0$;

$b_{41}(\tau+1) = t_{14}(\tau) * x_{24} / (0.5 + \|T_1 * X_1\|) = 0.4$;

Passo 8: Ainda existem padrões a serem apresentados. Voltar ao passo 2;

1	0	1	0
0	1	0	1
1	1	1	1

(a) Descendentes.

0.4	0	0.25
0	0.4	0.25
0.4	0	0.25
0	0.4	0.25

(b) Ascendentes.

Figura 2.9 – Matrizes de pesos da 2º iteração.

3º iteração ($\tau = 3$).

Passo 2:

$X_1 = [1\ 0\ 0\ 0]$;

Passo 3:

$y_j = 0$ (j=1:N);

Passo 4:

$y_1 = b_{11}(\tau) * x_{31} + b_{21}(\tau) * x_{32} + b_{31}(\tau) * x_{33} + b_{41}(\tau) * x_{34} = 0.4$;

$y_2 = b_{12}(\tau) * x_{31} + b_{22}(\tau) * x_{32} + b_{32}(\tau) * x_{33} + b_{42}(\tau) * x_{34} = 0.5$;

$y_3 = b_{13}(\tau) * x_{31} + b_{23}(\tau) * x_{32} + b_{33}(\tau) * x_{33} + b_{43}(\tau) * x_{34} = 0.25$;

Passo 5:

$y_1^* = 0.4$ (J =1).

Passo 6:

$\|X_1\|$; $\|T_1 * X_1\| = 2$; $\|T_1 * X_1\| / \|X_1\| = 1/1 = 1 > 0.7$;

X_3 (M_3) é categorizada sob o 1º nodo.

Passo7:

7.1: Atualização dos pesos descendentes (T). Ver Figura 2.10(a).

$t_{11}(\tau+1) = t_{11}(\tau) * x_{31} = 1$;

$t_{12}(\tau+1) = t_{12}(\tau) * x_{32} = 0$;

$t_{13}(\tau+1) = t_{13}(\tau) * x_{33} = 0$;

$t_{14}(\tau+1) = t_{14}(\tau) * x_{34} = 0$;

7.2: Atualização dos pesos ascendentes (B). Ver Figura 2.10(b).

$b_{11}(\tau+1) = t_{11}(\tau) * x_{31} / (0.5 + \|T_1 * X_1\|) = 0.67$;

$b_{21}(\tau+1) = t_{12}(\tau) * x_{32} / (0.5 + \|T_1 * X_1\|) = 0$;

$b_{31}(\tau+1) = t_{13}(\tau) * x_{33} / (0.5 + \|T_1 * X_1\|) = 0$;

$b_{41}(\tau+1) = t_{14}(\tau) * x_{34} / (0.5 + \|T_1 * X_1\|) = 0$;

Passo 8: Não existe mais padrões de entrada. Fim.

1	0	0	0
0	1	0	1
1	1	1	1

(a) Descendentes.

0	0	0.25
0	0.4	0.25
0.4	0	0.25
0	0.4	0.25

(b) Ascendentes.

Figura 2.10 – Matrizes de pesos da 3º iteração.

O nodo 3 não foi utilizado e, portanto, duas células foram identificadas na matriz de incidência máquina-peça. Elas são: $C_1 = \{M_1, M_3\}$ e $C_2 = \{M_2\}$. Note que a utilização da ART1 não forma células de máquinas e famílias de peças simultaneamente. Um possível caminho para formar famílias de peças usando esta rede neural seria obter a transposta da matriz original (ou da matriz rearranjada, onde máquinas agrupadas juntas são colocadas em posições adjacentes) para repetir o processo que foi realizado para máquinas.

Entretanto, o número de famílias formadas deve ser igual ao número de células de máquinas e, para isto, um valor apropriado de ρ deve ser encontrado. Este procedimento é reportado por Dagli e Huggahalli [36].

Uma importante limitação deste algoritmo deve ser enfatizada: o resultado final do agrupamento é altamente dependente da ordem de apresentação dos padrões de entrada.

Por exemplo, se o padrão X_3 fosse apresentado primeiro, e X_1 em segundo (ou mesmo em terceiro), as duas máquinas M_3 e M_1 seriam agrupadas em células separadas (todos os nodos seriam utilizados portanto). Isto ocorreria devido ao fato do padrão X_3 que, por possuir somente 1 elemento não-nulo, força o vetor exemplar T_1 a contrair prematuramente. Isto é, vários bits de T são ajustados para zero através da operação de multiplicação tomada entre T e X_3 (passo 7.1 do algoritmo ART1). Como X_1 não será considerado similar a T_1 e ainda há memória (nodos) disponível, então X_1 será categorizado por um nodo ainda não utilizado.

O problema de contração de vetores exemplares utilizando a ART1 no modo de aprendizado rápido, aplicada ao problema de agrupamento de máquinas e/ou peças, será discutido com maiores detalhes na seção seguinte.

2.6.1.1 APLICAÇÃO DA ART1 NO AGRUPAMENTO DE MÁQUINAS E PEÇAS

A performance da rede neural ART1 aplicada a formação de células de máquinas foi investigada por diversos pesquisadores [36, 37, 38, 39, 40, 41, 42, 43, 44]. Conforme a constatação de muitos destes pesquisadores [35, 36, 40, 43], a ordem de apresentação dos vetores à rede ART1 afeta a performance de agrupamento da mesma (qualidade da solução) e, na presença de matrizes de incidência com elementos excepcionais (mal-estruturadas), diferentes ordens de apresentação podem conduzir a diferentes resultados finais. Nesta seção, alguns trabalhos reportados na literatura que melhoram a performance da ART1 quando aplicada aos problemas de agrupamento de máquinas e/ou peças são apresentadas.

Kaparthi e Suresh [38] aplicaram uma rede neural ART1 para categorizar matrizes de incidência de grandes dimensões, envolvendo 10.000 peças e 100 tipos de máquinas. Dois conjuntos de dados foram gerados, um com uma perfeita estrutura diagonal de blocos (5 blocos diagonais de iguais dimensões, contendo cada um 2000 peças e 20 tipos de máquinas) e outra com imperfeições (8 blocos diferentes).

Três conjuntos de dados, com dimensões bem menores obtidos da literatura, foram também resolvidos e comparados pelos autores.

Seus experimentos computacionais revelaram que, quando uma estrutura diagonal de blocos existe no conjunto de dados original, a rede ART1 é capaz de identificar a estrutura com grande eficiência, até mesmo quando existem algumas imperfeições nos dados.

Eles observaram ainda que a rede neural ART1 foi mais precisa para identificar grupos quando a densidade dos elementos com valor igual a 1 na matriz de incidência não é muito baixa (ela normalmente é em matrizes de incidência). Para contornar este problema e realizar seus experimentos, eles reverteram os zeros e uns da matriz de incidência no processo de agrupamento e restauraram aos valores originais após a categorização.

A utilização da notação reversa reportada por Kaparthi e Suresh [38] foi justificada em [39] e a robustez da ART1 modificada foi demonstrada. Os autores definiram a robustez de uma rede neural ART1 como sendo a habilidade da rede em gerar soluções consistentes independente da seqüência de vetores apresentados a ela.

Admitindo que um vetor de entrada X foi considerado similar a um vetor exemplar T, de acordo com um determinado limite de vigilância ρ, os 4 casos a seguir mostram que a notação reversa pode melhorar a eficiência de agrupamento da rede neural ART1 [39]:

- **Caso 1:** O vetor de entrada X é idêntico ao vetor exemplar T. Neste caso, T permanece igual.

- **Caso 2:** O vetor X é diferente de T, a norma de T é menor do que a norma de X (a norma para um vetor binário é igual ao número de elementos não-nulos do vetor) e T está totalmente contido em X. Por exemplo, considere T = [1 0 0 1 1 0 0] e X = [1 1 0 1 1 0 0]. Observando estes vetores, nota-se que eles representam este segundo caso. Aqui, é desejável que T absorva os elementos não-nulos de X (neste exemplo, o segundo elemento de X). Utilizando a notação reversa, os dois vetores T e X seriam escritos como: T^r = [0 1 1 0 0 1 1] e X^r = [0 0 1 0 0 1 1]. Executando o passo 7.1 do algoritmo ART1, o vetor T^r atualizado seria igual a [0 1 1 0 0 1 1], o que representa retornando à notação original, T = [1 1 0 1 1 0 0]. Este vetor exemplar possui o segundo elemento não-nulo (ele absorveu o segundo elemento não-nulo de X como desejado, o que não ocorreria usando a notação original, pois o vetor T atualizado permaneceria o mesmo).

- **Caso 3:** O vetor X é diferente de T, a norma de T é maior do que a norma de X e X está totalmente contido em T. Neste caso, é desejável que T não contraia, isto é, não perca elementos não-nulos no processo de atualização. Por exemplo, considere T = [1 1 0 1 1 0 0] e X = [1 0 0 1 1 0 0]. A notação reversa para estes dois vetores é: T^r = [0 0 1 0 0 1 1] e X^r = [0 1 1 0 0 1 1]. A atualização do vetor T^r (passo 7.1 do algoritmo ART1) fornece T^r = [0 0 1 0 0 1 1], o que na notação original significa T = [1 1 0 1 1 0 0]. Observe que o vetor T não perdeu nenhum elemento não-nulo no processo de atualização. Isto não ocorreria utilizando a notação original, pois forneceria um vetor contraído T = [1 0 0 1 1 0 0].

- **Caso 4:** O vetor X é diferente de T e a norma de X é igual a norma de T. Neste caso, é desejável que o vetor T absorva os novos elementos não-nulos de X sem que T perca qualquer elemento não-nulo no processo de atualização. Por exemplo, considere T = [1 1 0 1 1 0 0] e X = [1 0 1 1 1 0 0]. A notação reversa é: T^r = [0 0 1 0 0 1 1] e X^r = [0 1 0 0 0 1 1]. O vetor exemplar é atualizado para T^r = [0 0 0 0 0 1 1], o que segundo a notação original é igual a T = [1 1 1 1 1 0 0] (o vetor exemplar absorveu o terceiro elemento não-nulo de X). Se somente a notação original fosse usada, o vetor T atualizado seria igual a T = [1 0 0 1 1 0 0] (ocorrendo uma contração do vetor).

Segundo os casos apresentados acima, a notação reversa previne a contração dos vetores exemplares, a qual se constitui como um dos maiores problemas da rede neural ART1. Esta contração ocorre devido à operação de multiplicação do passo 7.1 do algoritmo ART1 (operação lógica E), entre os vetores T e X. Uma vez que um determinado bit de T é ajustado para zero, ele nunca mais poderá ser ajustado novamente para 1 (lógico, qualquer multiplicação por zero será sempre zero). Conseqüentemente, este vetor T prematuramente contraído (em números de uns) poderá conduzir a classificações incorretas nas apresentações dos vetores de entrada restantes. Kaparthi e Suresh [35, 39] proporam mudanças dos passos do algoritmo ART1 original (ver Figura 2.11) para permitir a utilização da notação reversa nos vetores binários obtidos a partir da matriz máquina-peça.

Passo 2: *Aplicar um padrão de entrada binário X em notação reversa à rede neural ART1 (ler zeros como uns e uns como zeros);*

Passo 6: *Teste de Vigilância:*

Φ = *número de zeros nas posições em comum entre os vetores T e X;*

Π = *número de zeros do vetor de entrada X;*

Se $\left\{ \Phi / \Pi \right\} \leq \rho$ *então* $\{ y_J = -1 \}$ *(inibir o nodo J} e retorne para o passo 4, senão continue;*

Figura 2.11 – Mudanças do algoritmo ART1 original para acomodar a notação reversa proposta por Kaparthi e Suresh [35, 39].

Chen e Cheng [43] discutiram sobre o problema da ART1 e propuseram um conjunto de procedimentos suplementares (algoritmos de rearranjo e re-atribuição) para melhorar a performance da ART1 na presença de matrizes mal-estruturadas. Os autores argumentaram que a heurística reportada por Kaparthi e Suresh em [35, 38, 39] nem sempre conduz a uma solução razoável (eles ilustraram com um exemplo) e mesclaram sua heurística com aquela proposta por Kaparthi e Suresh, reportando bons resultados.

Chen et. al. [40] propôs uma heurística interessante: fazer a rede ART1 trabalhar com valores discretos +1 e –1 (o valor zero utilizado pela rede original seria substituído por este). Dagli e Huggahalli [36] também propuseram algumas heurísticas para melhorar a performance da rede ART1 quando aplicada ao problema de formação de células de máquinas.

2.6.2 FUZZY ART (FART)

Suresh e Kaparthi [35] investigaram a performance da rede neural *Fuzzy* ART (FART) para o agrupamento de máquinas e peças. Esta rede neural foi comparada com os algoritmos ROC2, DCA e as redes ART1 e ART1/KS. Segundo os teste realizados pelos autores, a FART foi mais eficiente na identificação de estruturas diagonais de blocos sendo, portanto, uma alternativa viável para o problema de agrupamento máquina-peça. O algoritmo FART é apresentado a seguir [35].

Algoritmo FART

Entrada: $[a_{ij}]_{m*n}$ ***(Matriz inicial)***

Saída:

Passo 1: Ajustar $\tau = 1$*; Inicializar os pesos das conexões* $w_{ji}(\tau) = 1$ $(i=1:M; j=1:N)$*;*

Passo 1.1: Selecionar o parâmetro de escolha (α)*, a taxa de aprendizado* (β) *e o limite de vigilância* (ρ)*:* $\alpha > 0;\ 0 \leq \beta \leq 1;\ 0 \leq \rho \leq 1;$

Passo 2: Aplicar um padrão de entrada X à rede neural FART:

Passo 3: Calcular a função de escolha (T_j) *para cada nodo* $j \in F_2$*:* $(j=1:N)$

$$T_j = \left\| X \wedge W_j \right\| \Big/ \left(\alpha + \left\| W_j \right\| \right);$$

Onde $\wedge$ *é o operador difuso E, definido como* $(U \wedge Z) = min(u_i, z_i)$*.*

Passo 4: Selecionar o nodo j vencedor (T_j^*)*:*

$$T_j^* = \max_j \{T_j\},\ J = j;$$

Passo 5: Teste de Vigilância:

Se $\left\{ \left[\left\| X \wedge W_J \right\| \Big/ \left\| X \right\| \right] \leq \rho \right\}$ *então* $\{T_J = -1\}$ *(inibir o nodo J} e retorne para o passo 4, senão continue;*

Passo 6: Atualizar os pesos para o nodo J:

$$W_j(\tau + 1) = \beta \cdot (X \wedge W_J(\tau) + (1 - \beta) \cdot W_J(\tau);$$

Passo 7: Se {ainda existe padrões de entrada} então {habilite qualquer nodo j desabilitado no passo 5 e retorne ao passo 2}, senão pare.

Exemplo 4: Considere a matriz de incidência máquina-peça mostrada na figura 2.12 O uso do algoritmo FART para encontrar células de máquinas nesta matriz é apresentado a seguir.

	P_1	**P_2**	**P_3**	**P_4**
M_3	**1**			
M_1	**1**		**1**	
M_2		**1**		**1**

Figura 2.12 – Matriz de incidência máquina-peça.

1º iteração ($\tau = 1$).

Passo 1:

N=3; M = 4; $\alpha = 0.5$; $\beta = 0.1$; $\rho = 0.7$;

Passo 2:

$X_1 (M_3) = [1\ 0\ 0\ 0]$;

Passo 3:

$T_1 = \| X_1 \wedge w_1 \| / (\alpha + \| w_1 \|) = 1/(0.5+4) = 0.22$;

$T_2 = \| X_1 \wedge w_2 \| / (\alpha + \| w_2 \|) = 1/(0.5+4) = 0.22$;

$T_3 = \| X_1 \wedge w_3 \| / (\alpha + \| w_3 \|) = 1/(0.5+4) = 0.22$;

Passo 4: No caso de empate, escolher o nodo de menor índice (critério de desempate):

J = 1;

Passo 5:

$\| X_1 \wedge w_1 \| / \| X_1 \| = 1/1 = 1 > 0.7$;

$X_1 (M_3)$ é categorizado sob o 1º nodo;

Passo 6: Atualização dos pesos W_1 (ver Figura 2.14).

$w_{11}(\tau+1) = \beta*[\min(X_{11}, w_{11}(\tau))] + (1-\beta)* w_{11}(\tau) = 1$;

$w_{21}(\tau+1) = \beta*[\min(X_{11}, w_{21}(\tau))] + (1-\beta)* w_{21}(\tau) = 0.9$;

$w_{31}(\tau+1) = \beta*[\min(X_{11}, w_{31}(\tau))] + (1-\beta)* w_{31}(\tau) = 0.9$;

$w_{41}(\tau+1) = \beta*[\min(X_{11}, w_{41}(\tau))] + (1-\beta)* w_{41}(\tau) = 0.9$;

1	1	1
0.9	1	1
0.9	1	1
0.9	1	1

Figura 2.13 – Matriz de pesos atualizada (1º iteração).

Passo 7: Ainda existem padrões a serem apresentados. Voltar ao passo 2;

2º iteração ($\tau = 2$).

Passo 2:

$X_2 (M_1) = [1\ 0\ 1\ 0]$;

Passo 3:

$T_1 = \| X_2 \wedge w_1 \| / (\alpha + \| w_1 \|) = 1.9/(0.5+3.7) = 0.45;$

$T_2 = \| X_2 \wedge w_2 \| / (\alpha + \| w_2 \|) = 2.0/(0.5+4) = 0.44;$

$T_3 = \| X_2 \wedge w_3 \| / (\alpha + \| w_3 \|) = 2.0/(0.5+4) = 0.44;$

Passo 4: Selecionar o nodo vencedor (J): $T_1 = 0.45$ (J = 1);

Passo 5:

$\| X_2 \wedge w_1 \| / \| X_2 \| = 1.9/2 = 0.95 > 0.7;$

X_2 (M_1) é categorizado sob o 1º nodo;

Passo 6: Atualização dos pesos W_1 (ver Figura 2.14).

$w_{11}(\tau+1) = \beta*[\min(X_{21}, w_{11}(\tau))] + (1-\beta)* w_{11}(\tau) = 1;$

$w_{21}(\tau+1) = \beta*[\min(X_{21}, w_{21}(\tau))] + (1-\beta)* w_{21}(\tau) = 0.81;$

$w_{31}(\tau+1) = \beta*[\min(X_{21}, w_{31}(\tau))] + (1-\beta)* w_{31}(\tau) = 0.9;$

$w_{41}(\tau+1) = \beta*[\min(X_{21}, w_{41}(\tau))] + (1-\beta)* w_{41}(\tau) = 0.81;$

1	1	1
0.81	1	1
0.9	1	1
0.81	1	1

Figura 2.14 – Matriz de pesos atualizada (2º iteração).

Passo 7: Ainda existem padrões a serem apresentados. Voltar ao passo 2;

3º iteração ($\tau = 3$).

Passo 2:

X_3 (M_2) = [0 1 0 1];

Passo 3:

$T_1 = \| X_3 \wedge w_1 \| / (\alpha + \| w_1 \|) = 1.62/(0.5+3.52) = 0.40;$

$T_2 = \| X_3 \wedge w_2 \| / (\alpha + \| w_2 \|) = 2.0/(0.5+4) = 0.44;$

$T_3 = \| X_3 \wedge w_3 \| / (\alpha + \| w_3 \|) = 2.0/(0.5+4) = 0.44;$

Passo 4: No caso de empate (T_2 e T_3) , escolher o nodo de menor índice (critério de desempate): J = 2;

Passo 5:

$\| X_3 \wedge w_1 \| / \| X_2 \| = 2/2 = 1 > 0.7$;

X_3 (M_2) é categorizado sob o 2º nodo;

Passo 6: Atualização dos pesos W_1 (ver Figura 2.15).

$w_{11}(\tau+1) = \beta*[\min(X_{31}, w_{11}(\tau))] + (1-\beta)* w_{11}(\tau) = 0.9$;

$w_{21}(\tau+1) = \beta*[\min(X_{31}, w_{21}(\tau))] + (1-\beta)* w_{21}(\tau) = 1$;

$w_{31}(\tau+1) = \beta*[\min(X_{31}, w_{31}(\tau))] + (1-\beta)* w_{31}(\tau) = 0.9$;

$w_{41}(\tau+1) = \beta*[\min(X_{31}, w_{41}(\tau))] + (1-\beta)* w_{41}(\tau) = 1$;

1	**0.9**	**1**
0.81	**1**	**1**
0.9	**0.9**	**1**
0.81	**1**	**1**

Figura 2.15 – Matriz de pesos atualizada (3º iteração).

Passo 7: Não existe mais padrões de entrada. Fim.

O resultado final do agrupamento é o seguinte: $C_1 = \{M_3, M_1\}$ e $C_2 = \{M_2\}$. O terceiro nodo da FART não foi utilizado. Este simples exemplo mostra que a FART é capaz de produzir melhores resultados do que a ART1 (sem modificações). Se esta matriz fosse aplicada à ART1, a apresentação do primeiro padrão (M_3) iria forçar a contração do vetor exemplar T de tal maneira que o próximo padrão (M_1) não seria considerado similar à M_3. A formação de células de máquinas considerando grandes dimensões (e.g. 2800x70) foram reportadas por Suresh e Kaparthi [35].

Suresh et al. [45] estudou o problema de identificação de famílias de peças com similares rotas de processamento usando a rede FART. Isto representou um importante passo nesta área de pesquisa, pois um relevante aspecto do chão-de-fábrica, i.e., a seqüência de operações de uma peça, foi levado em consideração. Os trabalhos anteriores na área simplesmente agrupavam uma matriz de incidência máquina-peça binária. Outro trabalho relevante sobre a aplicação da FART para resolver o PFCM é o de Kamal e Burke [46].

2.6.3 MAPAS AUTO-ORGANIZÁVEIS DE KOHONEN (*SELF-ORGANIZING MAP* - SOM)

A rede neural SOM, desenvolvida pelo professor Teuvo Kohonen [47, 48, 49], é talvez um dos modelos de redes neurais artificiais mais investigadas e reportadas na literatura. Este reconhecimento deve-se a sua simplicidade de implementação e a ampla variedade de problemas para o qual ela pode ser aplicada.

Estas aplicações incluem, apenas para citar algumas, problema do caixeiro viajante [50], reconhecimento da fala [49], exploração de dados [51] e formação de células de manufatura [42, 52, 53].

SOM é um modelo de rede neural não-supervisionado que mapeia padrões de entrada de dimensão N para um arranjo de nodos neurais com menor dimensão (dimensão M e N >> M), preservando a ordem topológica existente no espaço do padrão de entrada de uma maneira tão precisa quanto possível [48]

Existem duas camadas na rede neural SOM (ver Figura 2.16): a camada de entrada e a camada de Kohonen. Usualmente, os nodos neurais da camada de saída são arranjados em uma ou duas dimensões.

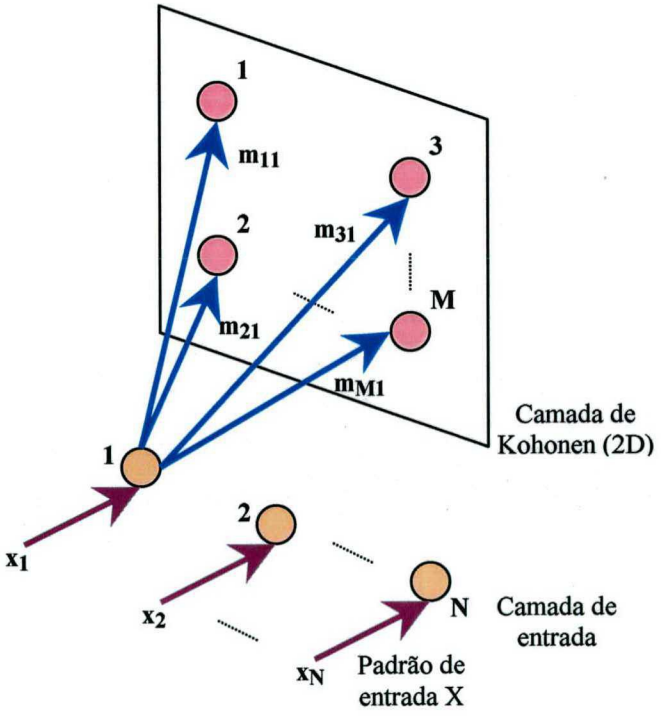


Figura 2.16 – Arquitetura típica de uma rede neural SOM (somente as conexões do nodo neural de entrada 1 foram mostradas a fim de facilitar a vizualização do leitor).

Usualmente, a camada de entrada contém N nodos neurais, equivalente à dimensão do padrão de entrada $X = \{x_1, x_2,..., x_N\}$, e a camada de Kohonen, também denominada de camada competitiva, contém M nodos neurais arranjados usualmente em um plano (2D). Estas camadas são totalmente interligadas através de conexões ponderadas.

Definindo os índices j e i para especificar os nodos neurais da camada de entrada e de Kohonen, respectivamente, o vetor peso de um nodo $i \in M$ é denotado por $\mathbf{m_i} = \{ m_{i1}, m_{i2},..., m_{iN}\}$, onde m_{ij} é o peso associado à conexão entre o nodo i da camada de Kohonen e o nodo j da camada de entrada.

Basicamente, para cada padrão de entrada $\mathbf{X} = \mathbf{X(t)} \in \Re^N$, sendo t é a coordenada do tempo, e um conjunto de vetores pesos $\mathbf{m_i(t)} \in \Re^N$ (inicializados de uma maneira apropriada), a idéia por trás do algoritmo da rede neural SOM é a seguinte. Em cada instante t, um padrão **X(t)** é comparado com todos os vetores $\mathbf{m_i(t)}$ de acordo com uma medida de similaridade, dado usualmente pela distância Euclideana entre os vetores $d(\mathbf{X},\mathbf{m_i})_E$. A distância Euclideana entre os vetores **X** e $\mathbf{m_i}$ é dada por:

$$d(X,m_i)_E = \|X - m_i\| = \left[\sum_{j=1}^{N} (x_j - m_{ij})^2\right]^{1/2} \quad (2.1)$$

Utilizando a equação (2.1), cada nodo neural i computa a medida de distância entre seu vetor peso $\mathbf{m_i(t)}$ e o padrão de entrada **X(t)** e, aquele com um valor $d(\mathbf{X},\mathbf{m_i})_E$ mínimo - $\mathbf{m_i(t)}$ mais similar à **X(t)** - é escolhido como o nodo vencedor. Este nodo é formalmente definido da seguinte forma:

$$\|X - m_i\| = \min_j \{\|X - m_i\|\}, \forall i = 1,2,...,M. \quad (2.2)$$

O nodo neural c (o índice c é utilizado para especificar o nodo vencedor) e seus vizinhos mais próximos, determinados por uma vizinhança $N_c(t)$, tem seus pesos atualizados e o padrão de entrada **X(t)** é atribuído ao grupo representado pelo nodo neural c. Desta forma, o vetor peso de um nodo neural i serve como um exemplar (ou vetor referência) dos padrões de entrada associados a este nodo [28].

A atualização dos pesos é sempre restrita a uma vizinhança N_c sendo dado por:

$$(2.3)\quad \begin{cases} m_i(t+1) = m_i(t) + h_{ci}(t)\cdot[X(t) - m_i(t)], & \forall i \in N_c(t); \\ m_i(t+1) = m_i(t), & \forall i \notin N_c(t). \end{cases}$$

Aqui, o termo $h_{ci}(t)$ é a função da vizinhança que decresce com o tempo e com a distância do nodo i a partir do nodo c. Ela define a região de influência que o padrão de entrada tem na rede neural SOM [50]. A função $h_{ci}(t)$ usualmente é utilizada sob duas formas: na primeira ela é constante sob toda a vizinhança do nodo c e zero em tudo mais; na segunda ela assume um formato de uma curva de sino (função gaussiana), dado por:

$$(2.4)\quad h_{ci}(t) = h_0(t)\cdot \exp\left(-\| r_c - r_i \|^2 \Big/ 2\sigma^2(t)\right)\cdot\alpha(t)\ ,$$

onde $\alpha(t) \in\ ]0,1[$ é uma função que decresce monotomicamente com o tempo, e r_c e r_i são as coordenadas dos nodos c e i, respectivamente.

O algoritmo da rede neural SOM pode ser descrito da seguinte forma [54].

Algoritmo SOM

Entrada: $[a_{ij}]_{m*n}$ *(Matriz inicial)*

Saída: $[a_{ij}]_{m*n}$ *(Matriz Final)*

Passo 1: Ajustar t= 0. Inicializar os pesos dos vetores pesos $\mathbf{m_i(t)}$ *aleatoriamente e definir o número máximo de iterações T.*

Passo 2: Aplicar um vetor $\mathbf{X(t)}$ *à rede neural SOM e computar a distância Euclideana* $d(\mathbf{X},\mathbf{m_i})_E$ usando a equação (2.1).

Passo 3: Encontrar o nodo mais similar ao padrão de entrada $\mathbf{X(t)}$ *usando a equação* (2.2).

Passo 4: Atualizar os vetores pesos $\mathbf{m_i(t)}$ *e calcular* $h_{ci}(t)$ *(se for o caso) usando as equações (2.3) e (2.4).*

Passo 5: Se $\{t \geq T\}$ *pare, senão incremente t e retorne ao passo 2.*

Exemplo 5: Considere a matriz de incidência máquina-peça do exemplo 4 (Figura 2.12).

O uso do algoritmo SOM para encontrar células de máquinas nesta matriz é apresentado a seguir. Nós iremos inicialmente fazer as seguintes suposições: (i) a camada de Kohonen contém 2 nodos dispostos em um *array* linear; (ii) como há somente dois nodos disponíveis, nós consideraremos que somente o nodo vencedor tem seus pesos atualizados; (iii) a taxa de aprendizado é dada por $\alpha(t+1) = 0.5*\alpha(t-1)$ e $\alpha(0) = 0.6$ [31].

1º Iteração (t=1):

Passo 1: T =3;

0.2	0.6	0.5	0.9
0.8	0.4	0.7	0.3

Figura 2.17 – Matriz de pesos $[m_{ij}]$ inicial (1º iteração).

Passo 2: $X_1 = [1\ 0\ 0\ 0]$;

$d(X_1, m_1) = [(X_{11}(t) - m_{11}(t))^2 + (X_{12}(t) - m_{12}(t))^2 + (X_{13}(t) - m_{13}(t))^2 + (X_{14}(t) - m_{14}(t))^2]^{1/2} = 1.44$;

$d(X_1, m_2) = [(X_{11}(t) - m_{21}(t))^2 + (X_{12}(t) - m_{22}(t))^2 + (X_{13}(t) - m_{23}(t))^2 + (X_{14}(t) - m_{24}(t))^2]^{1/2} = 0.883$;

Passo 3: $y_2 = 0.883$ (J = 2);

Passo 4:

$m_{21}(t+1) = m_{21}(t) + \alpha(0)*[X_{11}(t) - m_{21}(t)] = 0.92$;

$m_{22}(t+1) = m_{22}(t) + \alpha(0)*[X_{12}(t) - m_{22}(t)] = 0.16$;

$m_{23}(t+1) = m_{23}(t) + \alpha(0)*[X_{13}(t) - m_{23}(t)] = 0.28$;

$m_{24}(t+1) = m_{24}(t) + \alpha(0)*[X_{14}(t) - m_{24}(t)] = 0.12$;

$\alpha(t+1) = 0.5*\alpha(0) = 0.3$;

0.2	0.6	0.5	0.9
0.92	0.16	0.28	0.12

Figura 2.18 – Matriz de pesos $[m_{ij}]$ atualizada (1º iteração).

Passo 5: $t < T$, logo $t = t + 1 = 2$. Retornar ao passo 2;

2º Iteração (t=2):

Passo 2: $X_2 = [1\ 0\ 1\ 0]$;

$d(X_2, m_1) = [(X_{21}(t) - m_{11}(t))^2 + (X_{22}(t) - m_{12}(t))^2 + (X_{23}(t) - m_{13}(t))^2 + (X_{24}(t) - m_{14}(t))^2]^{1/2}$ $= 1.44$;

$d(X_2, m_2) = [(X_{21}(t) - m_{21}(t))^2 + (X_{22}(t) - m_{22}(t))^2 + (X_{23}(t) - m_{23}(t))^2 + (X_{24}(t) - m_{24}(t))^2]^{1/2}$ $= 0.752$;

Passo 3: $y_2 = 0.752$ $(J = 2)$;

Passo 4:

$m_{21}(t+1) = m_{21}(t) + \alpha(t-1)*[X_{21}(t) - m_{21}(t)] = 0.94$;

$m_{22}(t+1) = m_{22}(t) + \alpha(t-1)*[X_{22}(t) - m_{22}(t)] = 0.11$;

$m_{23}(t+1) = m_{23}(t) + \alpha(t-1)*[X_{23}(t) - m_{23}(t)] = 0.5$;

$m_{24}(t+1) = m_{24}(t) + \alpha(t-1)*[X_{24}(t) - m_{24}(t)] = 0.084$;

$\alpha(t+1) = 0.5*\alpha(t-1) = 0.15$;

0.2	0.6	0.5	0.9
0.94	0.11	0.5	0.084

Figura 2.19 – Matriz de pesos $[m_{ij}]$ atualizada (2º iteração).

Passo 5: $t < T$, logo $t = t + 1 = 3$. Retornar ao passo 2;

3º Iteração (t=3):

Passo 2: $X_3 = [0\ 1\ 0\ 1]$;

$d(X_3, m_1) = [(X_{31}(t) - m_{11}(t))^2 + (X_{32}(t) - m_{12}(t))^2 + (X_{33}(t) - m_{13}(t))^2 + (X_{34}(t) - m_{14}(t))^2]^{1/2}$ $= 0.678$;

$d(X_3, m_2) = [(X_{31}(t) - m_{21}(t))^2 + (X_{32}(t) - m_{22}(t))^2 + (X_{33}(t) - m_{23}(t))^2 + (X_{34}(t) - m_{24}(t))^2]^{1/2}$ $= 1.69$;

Passo 3: $y_1 = 0.678$ $(J = 1)$;

Passo 4:

$m_{11}(t+1) = m_{11}(t) + \alpha(t-1)*[X_{21}(t) - m_{21}(t)] = 0.17$;

$m_{12}(t+1) = m_{12}(t) + \alpha(t-1)*[X_{22}(t) - m_{22}(t)] = 0.66$;

$m_{13}(t+1) = m_{13}(t) + \alpha(t-1)*[X_{23}(t) - m_{23}(t)] = 0.43$;

$m_{14}(t+1) = m_{14}(t) + \alpha(t-1)*[X_{24}(t) - m_{24}(t)] = 0.92;$

$\alpha(t+1) = 0.5*\alpha(t-1) = 0.075;$

0.17	0.66	0.43	0.92
0.94	0.11	0.5	0.084

Figura 2.20 – Matriz de pesos $[m_{ij}]$
atualizada (3º iteração).

Passo 5: t = T, então pare.

O resultado final do agrupamento é o seguinte: $C_1 = \{M_3, M_1\}$ e $C_2 = \{M_2\}$. O mesmo procedimento pode ser feito para agrupar peças em famílias de peças, com a ressalva que o número de famílias de peças deve ser igual ao número de células de máquinas formadas.

2.7 AGRUPAMENTO DIFUSO

Algoritmos de agrupamento tentam organizar um conjunto de vetores de características não-rotulados em "grupos naturais" de maneira que os vetores que possuem características similares são agrupados juntos. Obviamente, aqueles que possuem características distintas são colocadas em grupos diferentes.

Algoritmos de agrupamento convencionais restringem que cada vetor pertença a exatamente um grupo, entretanto, podem existir vetores cuja atribuição a um grupo não seja uma tarefa fácil de determinar. Mais precisamente, certos vetores podem pertencer a vários grupos com diferentes graus de afinidade. Neste caso, o agrupamento difuso é uma abordagem mais adequada quando os dados disponíveis são imprecisos ou inexatos.

Considere um conjunto de n vetores $X = \{x_1, x_2,...,x_n \mid \forall k, x_k \in \Re^p\}$ para serem agrupados em c grupos ($2 \leq c < n$). O conjunto de todas as matrizes de partições difusas c*n é denotado por M_{fcn} e definido como [55, 56]:

$$(2.5)\ M_{fcn} = \left\{ U \in \Re^{c*n} \;\middle|\; \sum_{i=1}^{c} U_{ik} = 1;\ 0 < \sum_{k=1}^{n} U_{ik} < n;\ 0 \leq U_{ik} \leq 1;\ 1 \leq i \leq c;\ 1 \leq k \leq n \right\}.$$

Como $U_{ik} \in [0,1]$, um vetor $x_k \in X$ pode pertencer a diversos grupos ao mesmo tempo com diferentes graus de pertinência (pesos), pois no atual contexto, os grupos não são mutuamente exclusivos. O atual problema consiste em encontrar a matriz de partição difusa $U \in M_{fcn}$ que é ótima de acordo com um determinado critério de agrupamento.

Usualmente, o critério utilizado é dado por [57]:

$$(2.6)\ J_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n} (U_{ik})^m \cdot D_{ik}{}^2(v_i, x_k),$$

onde

$U \in M_{fcn}$; $m \in [1,\infty)$ é o coeficiente de ponderação (usualmente 2); $V = [v_1, v_2,...,v_c]$ é a matriz dos vetores referências (centróides) dos grupos e $D_{ik}(v_i, x_k)$ é uma medida da distância tomada entre x_k e v_i. Usualmente a distância Euclideana é empregada, i.e.,

$$(2.7)\ D_{ik}(v_i, x_k) = \left[\sum_{j=1}^{p} (x_{kj} - v_{ij})^2 \right]^{1/2}.$$

O vetor referência é dado por:

$$(2.8)\quad v_i = \left(\sum_{k=1}^{n} U_{ik}^{\ m} \cdot x_k\right) \Big/ \left(\sum_{k=1}^{n} U_{ik}\right).$$

A fim de eliminar as variáveis de U na equação (2.6) e trabalhar somente com as variáveis de V, nós podemos re-escrever a equação (2.6) utilizando a seguinte relação [58]:

$$(2.9)\quad U_{ik} = \frac{1}{\sum_{j=1}^{c} \left[D_{ik}(v_i, x_k)/D_{jk}(v_j, x_k)\right]^{2/(m-1)}},$$

para $m > 1$, $D_{jk}(v_j, x_k) > 0$ e para $1 \leq i \leq c$ e $1 \leq k \leq n$.

O novo funcional resultante é o seguinte:

$$(2.10)\quad R_m(V) = \sum_{k=1}^{n}\left[\sum_{i=1}^{c} D_{ik}^{\ 1/(1-m)}\right]^{1-m}.$$

A minimização da equação (2.10) permite implicitamente minimizar a equação (2.6). Bezdek e Hathaway [55] e Hall et al. [56] empregaram AGs para minimizar a equação (2.10).

Xu e Wang [58] utilizaram dados baseados nas características de projeto das peças para formar famílias de peças.

2.8 ALGORITMOS GENÉTICOS

Como os algoritmos genéticos (AGs) serão discutidos em detalhes no capítulo 4, nós vamos somente apresentar nesta subseção os trabalhos mais relevantes reportados na literatura cujo enfoque é a aplicação do AG na resolução do PFCM. O leitor não familiarizado com o tema em questão poderá consultar antes o capítulo 4 e retornar depois a esta subseção.

Venugopal e Narendran [8] foram os primeiros pesquisadores a utilizarem AGs para resolver o problema de formação de células de manufatura (PFCM). Eles utilizaram um AG para minimizar dois objetivos simultaneamente, a movimentação intercelular e a variação de carga dentro das células, empregando uma população para cada um dos objetivos. A representação inteira usada pelos autores é simples mas eficiente. O comprimento de um indivíduo é igual ao número total de máquinas, onde a posição de cada elemento do indivíduo corresponde a uma máquina. O valor contido nesta posição (máquina) é o número da célula para a qual esta máquina é atribuída (Figura 2.21). Esta representação inteira tem sido adotada por muitos pesquisadores [65, 66, 67, 70, 72, 80, 83].

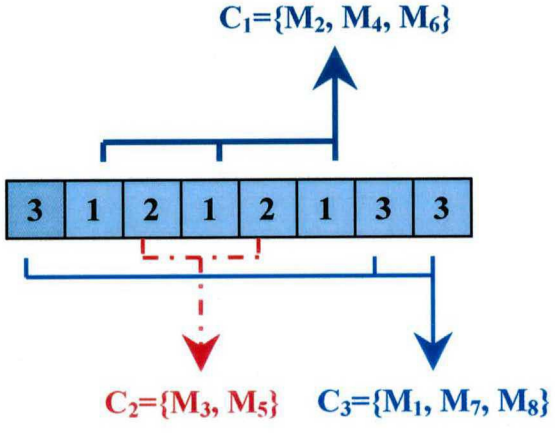


Figura 2.21 - 3 Células e 8 máquinas: cada gene do indivíduo representa uma máquina e seu valor a célula.

O estudo de Venugopal e Narendran considerou importantes fatores dos sistemas de manufatura, tais como a carga de trabalho em cada máquina, o tempo de processamento e demanda das peças.

Hon e Chi [59] formularam o problema de formação de famílias de peças como um modelo de programação inteira quadrática 0-1 para minimizar o número de peças excepcionais.

A efetividade da abordagem proposta foi comparada com sucesso em relação as outras técnicas utilizando 4 problemas obtidos da literatura. Essencialmente, os autores minimizaram os elementos excepcionais da matriz de incidência máquina-peça, sem levar em conta mais nenhum dado do ambiente de manufatura.

Kazerooni et al. [60, 61, 62] desenvolveram um novo coeficiente de similaridade tomado entre qualquer par de máquinas, denominado de similaridade de cadeia de máquinas (*machine chain similarity* – MCS), no qual toma em consideração as relações diretas ou indiretas entre qualquer um destes pares de máquinas. O coeficiente MCS utiliza uma faixa de valores igual a [0,1] (0 e 1 são valores extremos), onde para qualquer par de máquina (i, h) temos: se $MCS_{ih} = 1$, então toda a demanda de produção transportada no sistema está se movendo entre as máquinas i e h, direta ou indiretamente; se $MCS_{ih} = 0$, então nenhuma peça é movida entre as máquinas i e h, direta ou indiretamente. O coeficiente MCS é simétrico, isto é, $MCS_{ih} = MCS_{hi}$. O modelo matemático do coeficiente MCS desenvolvido pelos autores considera a demanda de produção e a seqüência de operações das peças.

Os autores empregaram um AG para obter uma matriz MCS otimizada, isto é, uma matriz ordenada na qual a similaridade de qualquer duas máquinas adjacentes é maximizada (máquinas perto uma das outras tem a mais alta similaridade), e um outro AG para maximizar a similaridade de peças adjacentes na matriz máquina-peça. Para o primeiro AG, a representação do indivíduo empregada indica a seqüência de máquinas que aparecem na matriz MCS (linhas ou colunas), enquanto que o segundo AG utiliza indivíduos que possuem um comprimento igual ao número de peças (N), e cada posição desta cadeia pode tomar um número a partir da faixa [0, N].

Kazerooni et al. [63] propôs uma abordagem integrada para o projeto de células de manufatura baseada em AGs. Eles consideraram 5 estágios, cada um sendo resolvido por um AG. Os três primeiros estágios são semelhantes àqueles reportados em [63]. No quarto estágio, células de manufatura são formadas e algumas máquinas para duplicação são escolhidas. Finalmente, no último estágio, o melhor leiaute intracelular e intercelular é obtido simultaneamente e o resultado é alimentado novamente ao estágio 4 para melhorar a configuração das células previamente formadas. Os estágios 4 e 5 são repetidos até que nenhuma melhora no projeto é atingida. Este trabalho considera não somente a formação de células de manufatura, mas também o projeto de leiaute das células (interno e externo). Trata-se, portanto, de um importante e completo trabalho dentro do atual contexto.

Entretanto, o tempo computacional pode ser demasiadamente alto para problemas complexos. Infelizmente, os autores não reportaram nenhuma informação sobre o tempo de computação consumido para resolver o exemplo contido no artigo.

Kazerooni et al. [64] estenderam seus trabalhos anteriores para levar em consideração a existência de rotas alternativas para as peças. Eles empregaram um AG para encontrar uma configuração de rotas de processamento que maximizassem o número de células de máquinas independentes (ou de forma análoga, minimizassem o número de peças excepcionais). Para fazer isto, eles maximizaram o número de elementos da matriz MCS que têm um valor zero ou abaixo de um certo limite determinado pelo usuário (L_m). Após determinar a configuração ótima de rotas, eles procederam como nos seus trabalhos anteriores.

Pereira [65] propôs um modelo matemático que minimiza a soma do custo anual do processamento das peças e o custo de amortização anual para investir numa máquina. O modelo tenta identificar as famílias de peças e células de máquinas simultaneamente, na presença de planos de processos alternativos, disponibilidade de mais de uma máquina do mesmo tipo e informações sobre as ferramentas. Informações adicionais incluem demanda, tempos e custos. Um exemplo da representação utilizada pela autora é apresentado na Figura 2.22.

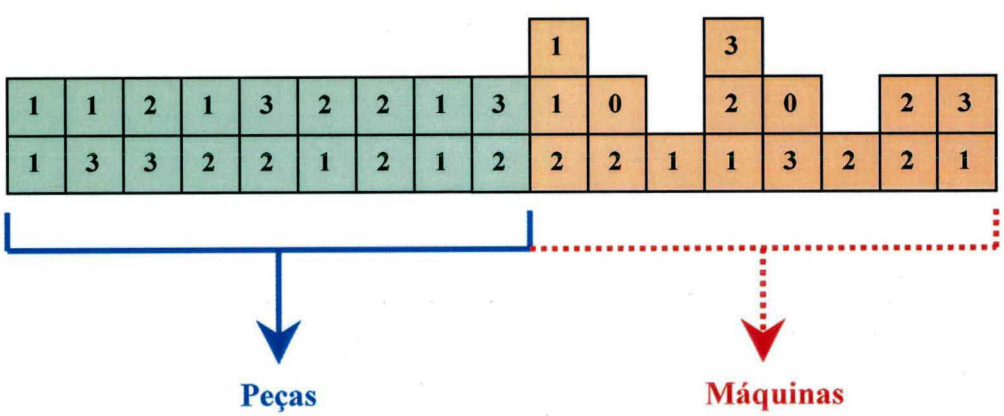


Figura 2.22 – Representação do indivíduo utilizado pelo AG proposto por Pereira [65].

Paris e Pierreval [66] formularam o problema de formação de células com o objetivo de minimizar o tráfego de trabalho em processo (WIP) intercelular global. Eles empregaram um algoritmo evolutivo distribuído para resolver o problema de minimização proposto.

A população é dividida em diversas sub-populações e cada uma delas é manipulada por um algoritmo evolutivo (AE). As subpopulações são dispostas em uma topologia do tipo hipercubo e indivíduos são trocados entre elas, segundo uma taxa de migração. Isto serve para promover a diversidade genética entre as subpopulações (isto é especialmente interessante quando alguma subpopulação converge para um ótimo local). Cada AE local possui seus próprios parâmetros, tais como probabilidade de recombinação, mutação, dimensão da população, taxa de migração e número máximo de gerações. O AE distribuído foi implementado em uma rede de estações de trabalho SUN® e testado em um problema hipotético elaborado pelos próprios autores (com um mínimo global conhecido).

Gupta et al. [67] reportaram um estudo que considerou aspectos realistas do chão-de-fábrica, tais como a carga de trabalho em cada máquina induzida por várias peças, a seqüência de operações e o impacto do leiaute das células na avaliação da movimentação intercelular e intracelular. O trabalho dos autores estendeu o estudo reportado em [8], além de apresentar um estudo estatístico da performance do AG proposto para quatro parâmetros do AG (número de gerações, tamanho da população, probabilidade de recombinação e mutação) e do número de células. As análises estatísticas foram realizadas usando ANOVA (*analysis of variance*), MANOVA (*multi-variate analysis of variance*) e o teste de Duncan.

Os autores consideraram dois modelos matemáticos, um proposto por Logendran [68] (minimização dos movimentos intercelulares e intracelulares) e o outro proposto por Venugopal e Narendran [8, 9] (minimização da variação da carga de trabalho nas células).

Hwang e Sun [69] formularam o PFCM usando um modelo matemático de atribuição quadrática generalizado para minimizar a movimentação intercelular total. O AG proposto pelos autores utilizou um novo método de codificação/decodifição para as soluções: a representação da solução codifica uma permutação das máquinas existentes, cada uma sendo unicamente identificada por um número. Uma heurística "gulosa" (*greedy*) foi empregada para atribuição das máquinas às células (decodificação).

Joines et al. [70] empregaram um AG para formar células de máquinas e famílias de peças simultaneamente. Eles utilizaram uma função objetivo que tenta minimizar o número de elementos excepcionais e o número de zeros nos blocos diagonais. Eles modificaram 6 operadores de variação (recombinação e mutação) desenvolvidos por Michalewicz [71] para trabalhar com representação inteira (eles inicialmente foram desenvolvidos para a representação de ponto flutuante). Os operadores são: (i) recombinação: aritmética e simples; (ii) mutação: uniforme, não-uniforme, multi-não-uniforme e limitada.

A representação dos indivíduos utilizada pelos autores é a mesma de Venugopal e Narendran [8], exceto que eles incluíram uma parte destinada as peças (ver Figura 2.23).

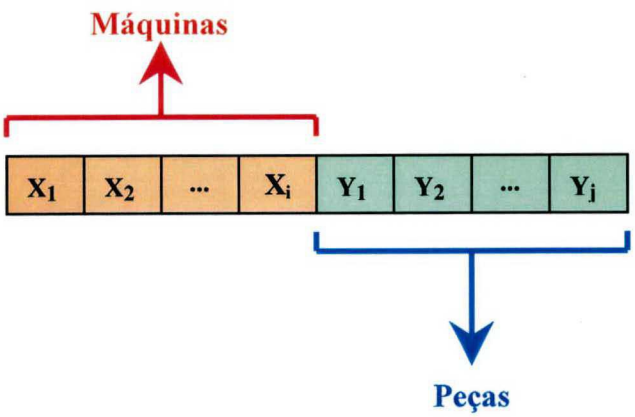


Figura 2.23 – Representação inteira do indivíduo proposto por Joines el al. [70].

Os valores de X_i e Y_j representam as células para os quais a máquina i e a peça j são atribuídas, respectivamente. Obviamente, X_i, $Y_j \in [1,L]$, onde L representa o número máximo de células permitido.

Zalzala et al.[72] empregaram um AG para minimizar uma função objetivo de soma ponderada envolvendo três objetivos: (1) minimização da movimentação intercelular, (2) minimização da variação da carga de trabalho dentro das células e (3) maximização da similaridade entre as máquinas. Os autores utilizaram o *software* Matlab® (versão 4.2) juntamente com o *toolbox* de AGs desenvolvido por Chipperfield et al. [73]. A representação dos indivíduos adotada é a mesma de Venugopal e Narendran [8].

Al-Sutan e Fedjki [74] propuseram um AG para resolver o problema de formação de família de peças. O modelo de programação inteira quadrática utilizado pelos autores foi originalmente formulado por Kusiak et al. [75]. A representação dos indivíduos utiliza o mesmo conceito daquele formulado em [8], exceto que peças são consideradas em vez das máquinas (e são atribuídas para famílias de peças em vez de células). Os autores testaram a sua abordagem em 14 problemas obtidos da literatura. A limitação deste trabalho é que os autores somente consideraram como dados de entrada a matriz de incidência máquina-peça, assim como fizeram Hon e Chi [59], realizando desta forma um agrupamento dos elementos não-nulos da matriz de incidência.

Joines et al. [76] desenvolveram um procedimento de busca local (PBL) baseado em duas regras de troca definidas por Ng [77] e incorporaram um AG.

Dada uma atribuição de máquinas e peças para células e famílias, respectivamente, a heurística utiliza um conjunto de regras para determinar a melhoria na função objetivo se a máquina i (peça j) foi trocada de sua célula (família) corrente para qualquer uma das L-1 células restantes. Isto é feito para todas as M máquinas (N peças).

Os autores discutiram sobre dois modelos básicos de evolução que têm sido usados para incorporar o "apredizado" dentro de um AG: o efeito Baldwin e a evolução Lamarckiana. Eles reportaram que o AG híbrido proposto possui melhor performance quando comparado com um AG sem busca local. Um conjunto de problemas (o maior contendo 115 máquinas e 2557 peças) foram utilizados para comparar diversas variações do AG híbrido. Este trabalho é uma extensão de Houck et al.[78], onde os autores apresentaram um estudo empírico que examinou o uso do aprendizado baseado na evolução Lamarckiana e no efeito Baldwin para três diferentes problemas (problemas de Corona, problema de alocação-localização e PFCM).

Souilah et al. [79] empregaram um AG para minimizar o tráfego intercelular com respeito a algumas restrições de co-habitação (máquinas que devem ser consideradas como uma só). Eles discutiram sobre alguns problemas relacionados com a utilização da representação de Venugopal e Narendran para resolver o PFCM e propuseram dois sistemas de codificação.

Gravel et al. [80] propuseram um método baseado em AGs para encontrar soluções não-dominadas de um problema de minimização bi-objetivo, onde cada peça possui 3 rotas alternativas de processamento. Os dois objetivos considerados pelos autores são a minimização do movimento intercelular e a variação da carga dentro das células.

Eles utilizaram os mesmo modelos formulados em [8], mas eles extenderam os modelos para considerar múltiplas rotas de processamento para os tipos de peças. A representação dos indivíduos é a mesma empregada em [8], mas ela inclui uma cadeia de números inteiros na qual cada posição da cadeia é uma peça e o valor contido nesta posição (peça) especifica a rota de processamento atribuída para a peça. Esta representação é semelhante aquela utilizada em [70]. Uma interessante particularidade deste trabalho é que os autores utilizaram dois AGs aninhados: o AG mais externo (denominado laço externo) busca encontrar a melhor atribuição de máquinas para as células, enquanto que o AG interno (denominado laço interno) busca determinar a melhor rota para as peças.

Os resultados produzidos no laço externo são avaliados de acordo com o melhor conjunto de rotas que podem ser encontrado no laço interno. A abordagem dos autores, entretanto, não garante que as soluções são realmente não-dominadas, além é claro, que a utilização de dois AGs aninhados pode acarretar em um tempo computacional muito elevado para problemas de grandes dimensões.

Cheng et al. [81] formulou o PFCM como um problema do caixeiro viajante (PCV) e propôs uma abordagem baseada em AGs para resolver o problema. Uma matriz de incidência máquina-peça binária é utilizada como dado de entrada. O problema de rearranjar colunas e linhas é similar ao problema de permutação e, desta forma, uma medida de distância para definir a similaridade entre pares de linhas (máquinas) e pares de colunas (peças) foi utilizado para determinar a matriz permutada final. O problema é resolvido colocando máquinas que processam peças similares perto uma das outras na permutação final. As peças colocadas perto uma das outras são aquelas que requerem processamento em máquinas similares. A implementação do AG dos autores é baseado no programa GENITOR desenvolvido por Whitley [82].

Pierreval e Plaquin [83] aplicaram o método NPGA (*Niched Pareto Genetic Algorithm*) desenvolvido por Horn e Nafpliotis [84, 85] para encontrar uma partição de 12 células de máquinas, com no máximo 5 máquinas por célula, a partir de um conjunto contendo 59 máquinas no total. Os dados foram obtidos de uma fábrica real que produz facas. Dois objetivos foram considerados pelos autores: a minimização do tráfego intercelular total e a minimização do desbalanceamento das células (diferença tomada entre as células que possuem a maior e menor carga para uma dada partição). Os objetivos são minimizados simultaneamente e um conjunto de soluções não dominadas (ver capítulo 3) foi obtido pelos autores. Eles apresentaram também um interessante operador de recombinação baseado nos seguintes princípios: (i) se duas máquinas pertencem à mesma célula em ambos os ancestrais, então estas máquinas serão atribuídas à mesma célula nos descendentes; (ii) o operador tentará privilegiar o agrupamento de máquinas que estão na mesma célula em uma das configurações dos ancestrais de forma aleatória. A representação dos indivíduos é a mesma de [60].

Su e Hsu [86] utilizaram uma abordagem híbrida envolvendo TS e AG para resolver um problema de minimização multi-objetivo. Os objetivos consistem em minimizar: (1) custos totais, que incluem o custo de transporte das peças (intercelular e intracelular) e o custo de investimento das máquinas; (2) desbalanceamento da carga de trabalho intrace-

lular e (3) desbalanceamento da carga de trabalho intercelular. O estudo dos autores considera a seqüência de operações, tempo de preparo e de operação, capacidade das máquinas, leiaute das células, duplicação de máquinas etc.

A solução é representada por um vetor que tem duas partes: a primeira é semelhante à representação de [8], mas ela também inclui múltiplas máquinas do mesmo tipo; a Segunda parte é utilizada para representar como as células são arranjadas no leiaute físico. As Figuras 2.24 e 2.25 ilustram um exemplo de uma solução codificada utilizada pelo AG proposto pelos autores contendo 8 máquinas (M_2 e M_3 são idênticas) e 3 células na primeira e segunda parte do seu código, respectivamente.

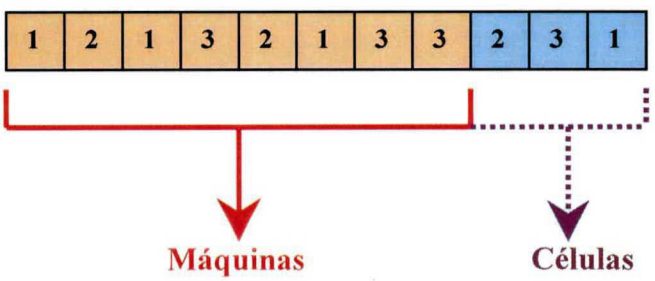


Figura 2.24 - Indivíduo utilizado pelo AG proposto por Su e Hsu [86].

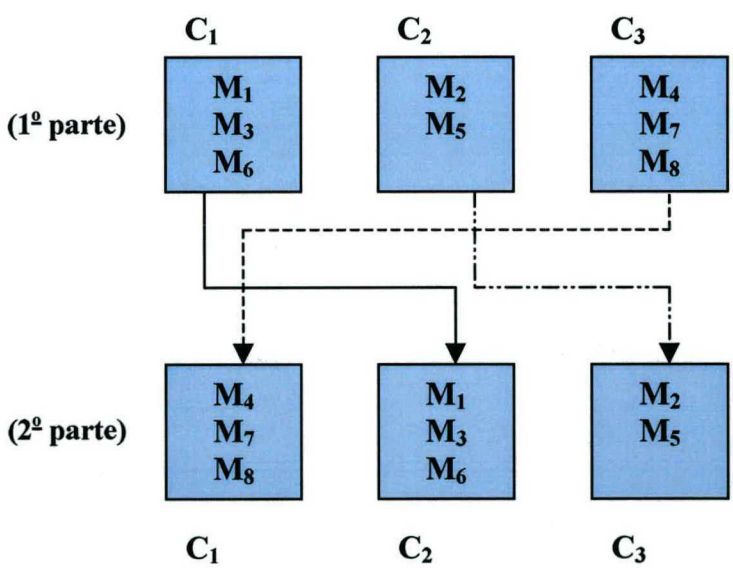
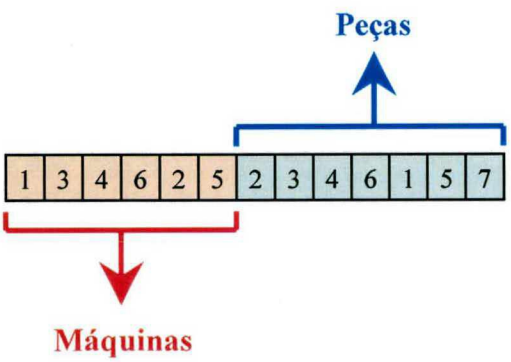


Figura 2.25 – Interpretação do indivíduo da Figura 2.24: a primeira parte do indivíduo (máquinas) especifica o agrupamento de máquinas; a segunda parte (células) determina como estes agrupamentos são arranjados fisicamente no leiaute.

Dimoupoulos e Mort [87] reportaram um estudo onde eles utilizaram um algoritmo de agrupamento hierárquico (SLCA) baseado em programação genética (PG) para diagonalização de matrizes de incidência binárias. O procedimento é baseado na evolução de um coeficiente de similaridade para cada problema considerado: quatro parâmetros de entrada são manipulados para construir uma fórmula do coeficiente de similaridade através do uso das 4 operações da aritmética básica. A formula é utilizada para o cálculo da matriz dos coeficientes de similaridade.

Mak et al. [88] propuseram um AG adaptativo para formar células de manufatura através do arranjo sucessivo das linhas e colunas de uma matriz máquina-peça binária **A** = $[a_{ij}]$, de forma que a medida da energia de ligação normalizada (α) da matriz **A** é maximizada. Quanto mais próximos os elementos não-nulos de **A** são colocados juntos, mais alto será o valor de α. Os autores empregaram uma representação simbólica para codificar a seqüência de máquinas e peças que aparecem em **A** (ver Figura 2.26): os primeiros M bits do indivíduo representam a seqüência de máquinas que aparecem nas linhas de **A**; os N bits restantes representam a seqüência de peças que aparecem nas colunas de **A**.



(a) Indivíduo com M + N bits: representação simbólica de **A** (Figura 2.26(b)).

	P_2	P_3	P_4	P_6	P_1	P_5	P_7
M_1		1	1				
M_3		1					
M_4		1	1				
M_6	1			1			
M_2							1
M_5					1	1	1

(b) Matriz de incidência máquina-peça **A** com 6 máquinas (M = 6) e 7 peças (N = 7).

Figura 2.26 – Representação utilizada em [88].

Na abordagem dos autores, as taxas de recombinação e mutação são ajustadas de forma adaptativa durante o processo de busca de acordo com a performance dos operadores genéticos. A efetividade da abordagem proposta foi testada em 13 problemas obtidos da literatura.

Plaquin e Pierreval [89] empregaram um AE para formar células de manufatura que levem em consideração as seguintes restrições: (1) algumas máquinas podem ter que ser agrupadas juntas ou porque elas irão compartilhar um recurso em comum (tais como gás, água etc.), ou porque elas requerem o conhecimento de um operador em particular; (2) algumas máquinas podem ter que ser separadas. Isto pode acontecer se certas máquinas causam interferências (tais como vibrações, altas temperaturas etc.) em outras máquinas e isto pode prejudicar estas de alguma forma.

Eles introduziram o conceito de agregado de máquinas: um conjunto de máquinas que devem estar juntas na mesma célula. As máquinas que são não vinculadas a nenhuma restrição do primeiro tipo são consideradas agregados unitários. A representação de um indivíduo é dada por um vetor de comprimento igual ao número de agregados de máquinas, onde o valor contido na posição deste vetor (agregado) representa a célula para o qual o agregado é atribuído. O objetivo do trabalho dos autores foi minimizar o tráfego intercelular.

De Lit et al. [90] propôs um AG específico para problemas de agrupamento, os quais ocorrem com freqüência no mundo real (o próprio problema de formação de células de manufatura, por exemplo). Eles argumentaram que os operadores genéticos padrões não são adequados para este tipo de problema.

Outros trabalhos relevantes nesta área são [91, 92, 93]. Dimopoulos e Zalzala [94] apresentaram uma excelente revisão dos trabalhos reportados na literatura cujo enfoque é a aplicação de AGs na manufatura, relatando, inclusive, o seu uso para resolver o PFCM.

Joines et al. [95] fez uma revisão da sua abordagem de AGs, desenvolvida em trabalhos anteriores [70, 76, 78], e mostrou que expressivos resultados podem ser obtidos, mesmo quando problemas de grandes dimensões são exigidos.

2.9 OUTRAS HEURÍSTICAS

Diversos autores formularam modelos matemáticos com o objetivo de otimizar um determinado critério e propuseram heurísticas diversas para resolvê-los [68, 96, 97, 98, 99, 100, 101].

Akturk e Balkose [101] propuseram uma abordagem multi-objetivo, a qual incluiu 6 funções objetivos a serem minimizados simultaneamente. Muitas outras referências podem ser obtidas em [3, 12, 102].

Têmpera simulada (TS) tem sido amplamente empregada para resolver o PFCM [97, 103, 104, 105, 106, 107, 108, 109, 110]. Vakharia e Chang [108] fizeram uma comparação entre TS e busca tabu (BT), concluindo que para o seu problema a TS foi melhor em termos de qualidade de solução e exigiu menos tempo computacional para resolver o problema.

2.9 PROGRAMAÇÃO MATEMÁTICA

Modelos de programação inteira linear (PIL) ou não-linear (PINL) são comumente utilizados para resolver o PFCM. A distinta vantagem destes modelos é a de poderem incorporar realísticos e importantes aspectos do chão-de-fábrica. Diversos autores propuseram modelos de PI que foram resolvidos através do uso de *softwares* comerciais, tais como LINDO®, LINGO® e MPSX/370® [98, 111]. A maioria dos *softwares* comerciais para resolver modelos de PIL 0-1 utiliza o método denominado *branch-and-bound* [98]. A principal desvantagem aqui é que a grande maioria dos modelos são computacionalmente intratáveis para os problemas reais das indústrias [12]. Para resolver problemas de grandes dimensões, é necessário recorrer a heurísticas.

2.10 CONCLUSÃO

Neste capítulo, diversas técnicas para resolver o PFCM foram apresentadas, muitas vezes acompanhadas com exemplos. O enfoque principal foi a inteligência computacional, principalmente redes neurais não-supervisionadas e algoritmos genéticos (AGs). Muitos trabalhos usando estas duas técnicas foram reportados durante a década de 90, sendo a primeira metade dedicada principalmente a aplicação das redes ART1, FART e SOM, e a segunda metade dedicada ao uso de AGs.

AGs representam uma importante e poderosa ferramenta para o projetista, pois eles podem otimizar um modelo matemático linear ou não-linear, podendo incorporar, inclusive, muitas informações do chão-de-fábrica.

CAPÍTULO 3

OTIMIZAÇÃO COMBINATÓRIA: CONCEITOS E HEURÍSTICAS

3.1 INTRODUÇÃO

Muitos problemas práticos encontrados em diferentes áreas, tais como engenharia, biologia, física etc., podem ser formulados como um problema de otimização envolvendo um ou mais objetivos.

A tarefa de otimizar um único objetivo consiste em empregar um método de otimização para localizar o ótimo global dentro de um espaço de busca discreto, contínuo ou misto, sujeito a um conjunto de restrições ou não.

A tarefa de otimizar um problema envolvendo múltiplos objetivos é muito mais complexa, pois estes objetivos podem ser conflitantes entre si e não-comensuráveis (medidos em unidades diferentes), devendo ser resolvidos simultaneamente. O conceito de solução para um problema de otimização multi-objetivo (POM) difere daquele atribuído para um problema envolvendo um único objetivo. Em vez de um único valor ótimo, há um conjunto de soluções alternativas, denominadas de soluções ótimas de Pareto ou não-dominadas. Quando todos os objetivos são considerados, não existe nenhuma solução no espaço de busca que é superior ao conjunto de soluções não-dominadas, e neste sentido, elas são consideradas ótimas. Em uma solução não-dominada nenhuma melhora pode ser obtida em um objetivo sem causar a simultânea degradação em ao menos um dos outros objetivos [112, 113].

Desde que nenhuma solução não-dominada pode ser considerada um ótimo absoluto, deve-se encontrar tantas soluções deste tipo quanto possível, para que um processo de tomada de decisão, baseado em preferências e de algum conhecimento específico, possa escolher uma solução.

Neste trabalho, nós estamos interessados em uma classe especial de problemas de otimização: a otimização combinatória. Esta classe de problemas trata com a minimização (ou maximização) de uma função objetivo (tipicamente a busca por um agrupamento, ar-

ranjo, ordenação ou seleção ótima de objetos discretos), na qual as variáveis de decisão podem somente admitir valores inteiros (tipicamente 0 e 1) para as soluções válidas.

Usualmente estes problemas possuem múltiplos mínimos locais e estão sujeitos a restrições que devem ser satisfeitas, tornando-os mais difíceis de serem resolvidos. Geralmente, métodos heurísticos são empregados para obter boas soluções, uma vez que problemas combinatórios (classe NP) podem ser intratáveis por métodos exaustivos.

No restante deste capítulo, o problema de otimização será primeiro formalmente definido, incluindo ambos os casos que envolvem um objetivo e múltiplos objetivos. Em seguida, serão discutidos alguns aspectos dos problemas combinatórios NP-completo e alguns métodos heurísticos para resolução destes problemas serão apresentados.

3.2 CONCEITOS BÁSICOS E TERMINOLOGIA

Definição 3.1 (Problema de otimização global): *Dado um espaço de busca discreto $\aleph$ e uma função objetivo f: $\aleph \rightarrow \Re$ (onde $\Re$ é o conjunto dos reais), a tarefa de otimização consiste em encontrar uma solução factível x^* que, sem perda de generalidade, minimize f, i.e.,*

$$\textit{Minimizar } \{f(x)\}$$
$$\textit{sujeito a } x \in \Gamma,$$

onde Γ é o conjunto de pontos factíveis do espaço de busca $\aleph$ ($\Gamma \subseteq \aleph$). □

Nós consideramos o problema de otimização como uma tarefa de minimização, o que não restringe a generalidade dos conceitos apresentados, uma vez que qualquer tarefa de maximização pode ser especificada como uma tarefa de minimização usando a seguinte relação:

$$\text{Maximizar } \{f\{x\} \mid x \in \Gamma\} = - \text{ Minimizar } \{- f\{x\} \mid x \in \Gamma\}$$

Definição 3.2 (Problema de otimização multi-objetivo - POM): *O POM consiste em encontrar o vetor de variáveis de decisão discretas $\boldsymbol{X}^* = (x_1^*, x_2^*, ..., x_n^*) \in \Gamma$ tal que o vetor objetivo $\boldsymbol{F(x)} = (f_1(x), f_2(x), ..., f_k(x))$ seja minimizado, i.e.,*

$$\text{Minimizar } \{\boldsymbol{F(x)} = (f_1(x), f_2(x), ..., f_k(x)) \mid \boldsymbol{X} \in \Gamma\}.$$ □

Definição 3.3 (Dominância de Pareto): *Dado um par de vetores* $\mathbf{u} = (u_1,...,u_k)$ *e* $\mathbf{v} = (v_1,...,v_k)$, $\mathbf{u}$ *é dito dominar* $\mathbf{v}$ *se, e somente se,* $\mathbf{u}$ *é parcialmente menor do que* $\mathbf{v}$ *(escrito como* $\mathbf{u} \prec \mathbf{v}$*), i.e.,*

$$\forall i \in \{1,...,k\},\ u_i \leq v_i \wedge \exists i \in \{1,...,k\} : u_i < v_i. \ \square$$

Definição 3.4 (Solução ótima de Pareto): *Uma solução* $\mathbf{x}_u \in \Gamma$ *é dita ser ótima de Pareto (ou não-dominada) se, e somente se, não existe nenhum* $\mathbf{x}_v \in \Gamma$ *tal que* $\mathbf{v} = f(\mathbf{v}) = (v_1,...,v_k)$ *domina* $\mathbf{u} = f(\mathbf{u}) = (u_1,...,u_k)$. $\square$

Considere o exemplo hipotético apresentado na Figura 3.1. Esta figura representa os valores da função objetivo f_1 graficados contra aqueles da função objetivo f_2 para o mesmo valor da variável independente. Repare que estamos minimizando ambos os objetivos (ao longo dos eixos x e y).

Os pontos da Figura 3.1 (A,B,C,D e E) representam as soluções não-dominadas no espaço objetivo e formam uma superfície conhecida como fronteira ótima de *Pareto* [84, 114]. A Figura 3.1 mostra estes pontos interpolados por uma possível função quadrática (linha pontilhada).

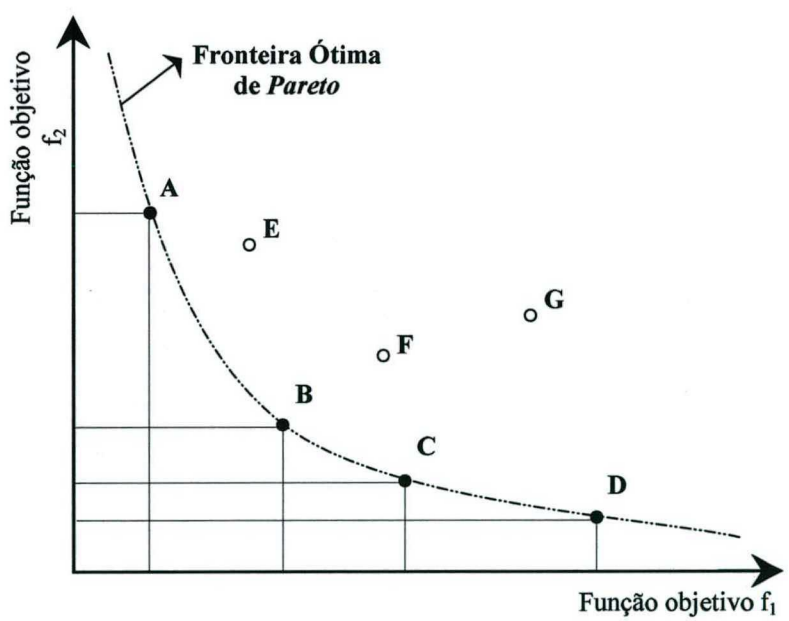


Figura 3.1 – Ilustração da fronteira ótima de Pareto.

O ponto A representa uma solução que fornece o menor valor para f_1, entretanto, ela resulta em um valor alto para f_2 (para a solução E o raciocínio é similar, mas inverso).

Repare que as soluções são parcialmente ordenadas: $f_1^A < f_1^B < f_1^C < f_1^D < f_1^E$ e $f_2^A > f_2^B > f_2^C > f_2^D > f_2^E$. Qual é a melhor solução contida na fronteira ótima de *Pareto*? Nós não podemos realmente responder esta questão, pois uma solução é melhor do que outra em um objetivo e pior no outro.

Os pontos brancos da Figura 3.1 (F e G) são denominados de soluções dominadas (ou inferiores). Embora não possamos afirmar nada quando comparamos F com {A, C, D, E} e G com {A, D, E}, existe uma solução B e C que são melhores do que F e G em ambos os objetivos, respectivamente. Neste caso, dizemos que B domina F e C domina G (a dominância é transitiva).

Como a tarefa de otimização de um vetor de funções objetivos (conflitantes) gera um conjunto muito grande de soluções não-dominadas, que representam compromissos (*trade-offs*) entre os diversos objetivos, torna-se necessário um processo de escolha baseado em preferências. Este processo é conhecido como tomada de decisão.

Um método clássico para resolver um POM consiste em combinar todos os objetivos em uma única função, i.e., fazer a soma ponderada dos objetivos a serem otimizados (usualmente, os pesos utilizados são normalizados). Resolver a função de soma ponderada para diferentes combinações de pesos rende um conjunto de soluções não-dominadas. Deve-se salientar que estes pesos não refletem a relativa importância dos objetivos uns em relação aos outros, mas, sim, a localização das soluções não-dominadas [112]. Entretanto, para um conjunto não-convexo, algumas soluções não-dominadas não podem ser encontradas para qualquer combinação dos pesos. Desta forma, este método é mais adequado para resolver problemas cuja superfície (conjunto) de soluções é convexo. Para outros métodos tradicionais de resolução, o leitor interessado pode consultar [112].

Desde a metade da década de 90, o emprego de métodos baseados em algoritmos evolutivos para resolver POM, denominados de algoritmos evolutivos multi-objetivos (A-EMOs) tem atraído uma grande atenção por parte dos pesquisadores e, em especial, da comunidade de AEs. Um trabalho pioneiro nesta área foi desenvolvido por David Shaffer em 1985, que implementou uma extensão de um AG simples e o denominou de VEGA (Vector Evaluated Genetic Algorithm). Apesar de seu valor histórico, a implementação de Shaffer possui conhecidos problemas.

Diversos autores reportaram o desenvolvimento de AEMOs, entre os quais podemos citar aqueles que freqüentemente são utilizados para resolver problemas práticos ou funções numéricas de teste:

- *Niched Pareto Genetic Algorithm* (NPGA): proposto por Horn e Nafpliotis [84, 85];
- *Nondominated Sorting Genetic Algorithm* (NSGA): proposto por Srinivas e Deb [115];
- *Multi-objective Genetic Algorithm* (MOGA): proposto por Fonseca e Fleming [113, 116, 117];
- *Strength Pareto Evolutionary Algorithm* (SPEA): proposto por Zitzler e Thiele [118, 119, 120].

Todos estes métodos utilizam o conceito ótimo de *Pareto* e empregam formas específicas de atribuição dos valores de aptidão para os indivíduos baseados neste conceito. O método SPEA, em especial, incorpora uma estratégia elitista.

3.3 PROBLEMAS NP-COMPLETOS

A classe mais importante e difícil dos problemas de otimização combinatória (POC's) são do tipo NP-completo (nondeterministic polynomial time complete), sendo de particular interesse deste trabalho. Problemas da classe NP-completo requerem uma quantidade de tempo de resolução, para se obter uma solução ótima exata, que cresce exponencialmente com a dimensão do problema [121]. Desta forma, métodos exatos, o qual garantem o ótimo global, não podem ser utilizados para problemas de grandes dimensões devido ao elevado tempo computacional requerido para resolvê-los, mesmo utilizando os mais potentes supercomputadores da atualidade.

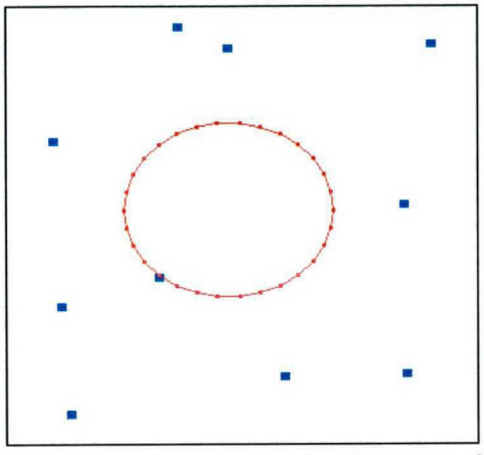
Se um POC NP-completo possui dimensão N, então as possíveis soluções são da ordem de exp(N) ou N! [122]. Um exemplo clássico desta classe de problemas é o problema do caixeiro viajante (PCV). Ele pode ser enunciado da seguinte forma:

> *Dado uma lista contendo N cidades e as distâncias entre cada cidade, um vendedor deve partir de uma cidade qualquer e passar por todas as cidades, uma única vez, retornando à cidade de sua partida. O problema consiste em minimizar a distância percorrida pelo vendedor.*

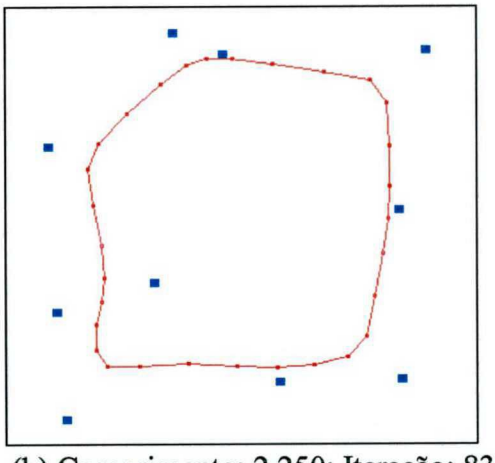
Nós podemos considerar que a distância de uma cidade n_1 até n_2 é a mesma que de n_2 até n_1, e, portanto, a distância percorrida pelo vendedor é a mesma nos dois sentidos do circuito (neste caso, o PCV é simétrico). Para este caso, existem (N-1)!/2 soluções distintas. Como pode ser observado, à medida que N cresce, torna-se rapidamente impraticável a busca exaustiva para o PCV (e para todos os problemas NP-completo).

Métodos heurísticos são freqüentemente empregados para obter uma boa solução factível (próxima da solução ótima) para problemas combinatórios em um tempo computacional aceitável. Entretanto, não há nenhuma garantia que estes métodos obtenham a solução ótima. Os métodos heurísticos não avaliam cada solução potencial do espaço de busca (exploração exaustiva), muito pelo contrário, eles exploram apenas um subconjunto deste espaço. Entretanto, baseados em determinados critérios, os quais são específicos do método empregado, eles conseguem direcionar suas buscas para as regiões mais promissoras do espaço de busca. Os métodos heurísticos incluem, entre outros, algoritmos evolutivos [123], redes neurais [122], busca tabu [125] e têmpera simulada [126].

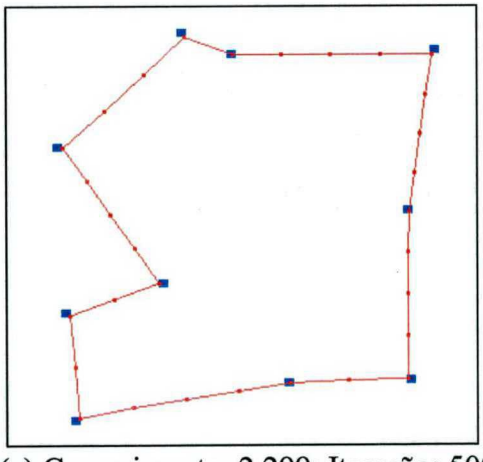
A Figura 3.2 ilustra o processo de resolução de um PCV com 10 cidades (distâncias tomadas aleatoriamente dentro de um plano 0-1) utilizando o método da rede elástica [127] (applet java obtido do *Dzhelepov Laboratory of Nuclear Problems*, Rússia).



(a) Comprimento: 1.254; Iteração: 0.



(b) Comprimento: 2.250; Iteração: 83.



(c) Comprimento: 2.299; Iteração: 500.

Figura 3.2 – Método da rede elástica para o PCV.

3.4 HEURÍSTICAS PARA OTIMIZAÇÃO COMBINATÓRIA

3.4.1 BUSCA GLOBAL E LOCAL

Nesta subseção, dois métodos opostos, mas que realizam uma busca cega, são apresentados: o método de Monte Carlo (busca aleatória) e o passo descendente. No primeiro (ver Figura 3.3), pontos no espaço de busca são gerados aleatoriamente de acordo com uma distribuição de probabilidade Ξ [128]. A busca é global e ocorre de forma totalmente aleatória, sem que nenhuma informação seja utilizada para guiar esta busca para as regiões mais promissoras do espaço de busca.

Algoritmo 1: Monte Carlo
Entrada:
f: $\aleph \rightarrow \Re$ *(função objetivo)*
Ξ *(distribuição de probabilidade)*
Saída:
$x \in \Gamma$ *(melhor solução encontrada)*

$i \leftarrow 0$;
$x \leftarrow$ *gerar_ponto*($\aleph$, Ξ);
enquanto $\{i < max\_iter\}$
    $x_a \leftarrow$ *gerar_ponto*($\aleph$, Ξ);
    se $\{f(x_a) - f(x) < 0\}$ ***então***
        $x \leftarrow x_a$;
        $i \leftarrow i + 1$;
    fim_se
fim_enquanto
retornar $\{x\}$

Figura 3.3 –Pseudo-código do método de Monte Carlo.

O método do passo descendente (ver Figura 3.4) realiza uma busca local, isto é, ele tenta melhorar a solução corrente x através da procura por uma solução x_v, obtida na vizinhança de x (Ω (x)), que permita reduzir o valor da função objetivo. Uma desvantagem

deste método é que como ele somente permite um movimento de "descida", ele não pode escapar de mínimos locais [129].

Algoritmo 2: Passo Descendente

Entrada:

$x_0 \in \Gamma$ *(solução inicial)*

$f: \aleph \rightarrow \Re$ *(função objetivo)*

$\Omega : \Gamma \rightarrow 2^{\Gamma}$ *(função geradora da vizinhança)*

Saída:

$x \in \Gamma$ *(melhor solução encontrada)*

$i \leftarrow 0;$

$x \leftarrow x_0;$

enquanto $\{i < max\_iter\}$

$x_v \leftarrow gerar\_vizinho(\Omega, x);$

se $\{f(x_v) - f(x) < 0\}$ ***então***

$x \leftarrow x_v;$

$i \leftarrow i + 1;$

fim_se

fim_enquanto

retornar $\{x\}$

Figura 3.4 –Pseudo-código do método passo descendente.

3.4.2 TÊMPERA SIMULADA

Têmpera simulada (TS) é uma técnica de otimização estocástica muito empregada para resolver complexos POC's. Ela é inspirada no processo físico de resfriamento de materiais sólidos (*annealing*) a uma taxa controlada. Quando um metal é levado ao estado líquido e resfriado muito lentamente, permitindo que as moléculas do metal possam atingir o equilíbrio térmico em cada temperatura, o material pode alcançar uma estrutura cristalina com uma energia potencial mínima. Entretanto, se o resfriamento é realizado muito rapidamente, as moléculas do material não dispõem de tempo suficiente para se reorientarem numa estrutura cristalina, resultando em um material amorfo com um estado de energia potencial mais alto. Este conceito básico do processo físico é simulado para POC's, buscando atingir soluções quase-ótimas.

Um trabalho pioneiro é o de Metropolis et al. [130], que propuseram um método de Monte Carlo para encontrar um estado de equilíbrio térmico de um cristal quando a temperatura T é conhecida (fixa). Os trabalhos de Kirkpatrick et al. [126] e Cerny [131] estimularam o desenvolvimento teórico e a aplicação da TS em POC's.

Essencialmente, a têmpera simulada procede da seguinte forma: a partir de um estado inicial factível x_0 e de uma temperatura inicial T_0, o algoritmo TS (Figura 3.5) em cada temperatura repetidamente gera um novo estado factível x_v (estado vizinho ou pertubado) na vizinhança de x_c (estado corrente) e aceita este novo estado de acordo com a seguinte probabilidade de aceitação (critério de Metropolis):

$$PA = \begin{cases} 1, \text{se} \Delta f_{c,v} \leq 0; \\ \exp\left(-\Delta f_{c,v} / T\right), \text{caso contrário.} \end{cases}$$

onde $\Delta f_{c,v} = f(x_c) - f(x_v)$.

Quando $\Delta f_{c,v} > 0$, o estado x_v é aceito somente se

$$\min\left\{1, \exp\left(-\Delta f_{c,v} / T\right)\right\} \geq U(0,1),$$

onde U(0,1) é um número aleatório tomado a partir de uma distribuição uniforme entre zero e um.

A probabilidade desta aceitação limitada, isto é, se $\Delta f_{c,v} > 0$, é

$$\begin{cases} \lim_{T\to\infty} \exp\left(-\Delta f_{c,v}/T\right) = 1; \\ \lim_{T\to 0} \exp\left(-\Delta f_{c,v}/T\right) = 0. \end{cases}$$

Isto significa que, quando T é suficientemente alto, a grande maioria dos estados gerados (x_v) é aceita pela TS, o que permite a livre exploração do espaço de busca. Mas, à medida que T decresce para zero, de acordo com uma função decrescente determinada pelo usuário, a TS torna-se muito mais seletiva em aceitar um novo estado [129]. No final, somente os estados que efetivamente minimizam a função objetivo f são aceitos pela TS.

A TS difere de outros métodos de busca local clássicos uma vez que ela permite aceitar uma solução inferior obtida na vizinhança x_c. Este movimento de subida da encosta de uma superfície (descrita por f) permite a TS evitar a permanência em mínimos locais e a convergir para soluções quase-ótimas. Com o progresso das iterações e conseqüentemente a redução de T, como discutido acima, a probabilidade da TS para aceitar soluções inferiores diminui.

Para a efetiva implementação da TS, uma série de decisões devem ser tomadas antes pelo usuário. Primeiro, após a formulação do problema, deve-se responder as seguintes perguntas:

(1) O que é uma solução (estado factível) para o problema em questão?

(2) Como nós podemos determinar uma solução inicial?

Após responder estas questões, nós precisamos especificar uma função custo ou objetivo que meça a qualidade de um dado estado. Segundo, é preciso determinar a estrutura da vizinhança para o problema em questão. Isto é, como gerar um novo estado x_v na vizinhança de x_c? $V(x_c)$ representa o conjunto das soluções vizinhas do estado x_c obtidas através de uma transformação (ou movimento) elementar. Por convenção, nós consideramos que $x_c \notin V(x_c)$, $\forall x_c \in \Gamma$ e $V(x_c) \subseteq \Gamma$. Por último, e não menos importante como determinar o resfriamento do sistema de forma que uma boa solução possa ser encontrada no final pela TS. Aqui, quatro escolhas devem ser feitas: (1) temperatura inicial; (2) o comprimento da época, i.e. o número de iterações realizadas durante um mesmo valor de T; (3) uma regra para o decréscimo de T (atualização); (4) critério de parada. Este conjunto de decisões de-

termina a estratégia (ou escala) de resfriamento da TS [132]. A temperatura inicial T_0 deveria ser alta o suficiente para assegurar uma alta probabilidade inicial ($Pr_0 \in [0.8,0.9]$) a fim de aceitar quase todos os estados gerados em T_0. Em certas aplicações [133], esta probabilidade inicial foi selecionada como $Pr_0 = 0.4$. Se o valor de T estiver desnecessariamente acima do ideal, um elevado tempo computacional pode ser gasto pelo subseqüente processo de decréscimo de T.

Algumas funções são propostas na literatura para especificar o comprimento da época (ou *plateau*) L [134]:

(i) Constante: $L_k = Cte$, onde $Cte \in \Re^+$;

(ii) Aritmética: $L_k = L_{k-1} + Cte$;

(iii) Geométrica: $L_k = L_{k-1} / a$, onde $a \in (0,1)$;

(iv) Logarítmica: $L_k = Cte / \log(T_k)$;

(v) Exponencial: $L_k = (L_{k-1})^{1/a}$, onde $a \in (0,1)$.

A condição lógica dada no segundo laço do algoritmo TS (Figura 3.5) é determinado por L. Similarmente a L, as funções reportadas na literatura que determinam a forma de decremento da temperatura T são [134]:

(i) Aritmética: $T_k = T_{k-1} - Cte$;

(ii) Geométrica: $T_k = T_{k-1} * \alpha$, onde $\alpha \in (0,1)$. O parâmetro α é denominado taxa de resfriamento e é usualmente dado dentro do intervalo $0.85 \leq \alpha \leq 0.95$;

(iii) Inversa: $T_k = Cte / (1+k)$;

(iv) Logarítmica: $T_k = Cte / \log(1+k)$.

A função para o decréscimo de T comumente empregada na literatura é (ii).

Algoritmo 3: Têmpera Simulada

Entrada:

$x_0 \in \Gamma$ *(solução inicial)*

$T_0 \in \Re^+$ *(temperatura inicial)*

$f: \aleph \rightarrow \Re$ *(função objetivo)*

$\Omega : \Gamma \rightarrow 2^{\Gamma}$ *(função geradora da vizinhança)*

Saída:

$x_c \in \Gamma$ *(melhor solução encontrada)*

$T \leftarrow T_0$;

$x_c \leftarrow x_0$;

enquanto *{sistema não está congelado}*

    enquanto *{equilíbrio na temperatura T não é alcançada}*

        $x_v \leftarrow$ *gerar_vizinho*(Ω, x_c);

        se $\{f(x_v) - f(x_c) < 0\}$ ***então***

            $x_c \leftarrow x_v$;

        senão se $\{exp(f(x_v) - f(x_c))/T > U(0,1)\}$

            $x_c \leftarrow x_v$;

        fim_se

    fim_enquanto

    $T \leftarrow$ *reduzir_temperatura*(T);

fim_enquanto

retornar $\{x_c\}$

Figura 3.5 – Pseudocódigo da têmpera simulada.

3.4.3 ALGORITMOS EVOLUTIVOS

Algoritmos evolutivos (AEs) representam uma classe de métodos de otimização e busca estocásticos que são inspirados no processo de evolução natural das espécies.

A maioria das implementações correntes dos AEs deriva de três métodos-chaves [14]: algoritmos genéticos (AGs) [123], introduzidos por J. H. Holland; programação evolutiva (PE) [124], introduzido por L. J. Fogel e estratégia evolutiva (EE) [15]desenvolvida por I. Rechemberg e H.–P. Schwefel. Embora cada um destes três métodos implemente de uma forma particular os conceitos de seleção e genética natural, eles possuem uma estrutura similar em um nível de abstração mais alto (ver Figura 3.6).

Algoritmo 4: Algoritmo Evolutivo
Entrada:
N (tamanho da população)
Max_gen (número máximo de gerações)
$\Phi : I \rightarrow \Re$ *(função de aptidão)*
Saída:
$i \in P(t)$ *(melhor indivíduo da população);*

$t \leftarrow 0$;
$P(t) \leftarrow$ *criar_população(N);*
enquanto *{critério de parada == falso}*
    $\Phi(t) \leftarrow$ *avaliar_população(P(t));*
    $Q(t) \leftarrow$ *selecionar_população(P(t),* $\Phi(t)$*);*
    $Q(t) \leftarrow$ *variar_população(Q(t));*
    $P(t) \leftarrow Q(t)$;
    $t \leftarrow t + 1$;
fim_enquanto
retornar *{melhor indivíduo* $i \in P(t)$*}*

Figura 3.6 – Pseudocódigo do algoritmo evolutivo.

Basicamente, todos os AEs trabalham com uma população de indivíduos (um conjunto de potenciais soluções do espaço de busca) que são avaliados por uma função de aptidão e são modificados através de dois princípios da evolução: seleção e variação.

A população inicial pode ser gerada aleatoriamente de acordo com uma distribuição uniforme (típico) ou gerada utilizando algum conhecimento específico do problema em questão. O tamanho da população é geralmente considerado constante em um AE, embora isto não seja uma regra.

Essencialmente, o AE simula a evolução natural através de um processo de computação iterativo que utiliza os operadores de avaliação, seleção e variação (representados na Figura 3.6 pelas funções avaliar_população, selecionar_população e variar_população, respectivamente) em uma população durante um determinado número de gerações. Cada geração representa uma execução do laço finito do algoritmo 4. Sem perda de generalidade, um número máximo de gerações é definido como critério de parada do algoritmo 4, entretanto, outros critérios podem ser utilizados para terminar a simulação.

A função de aptidão fornece a cada indivíduo uma medida qualitativa com respeito ao problema considerado. Então, baseado nestes valores, a seleção possibilita que a busca seja direcionada para as regiões mais promissoras do espaço de busca por permitir que indivíduos mais aptos, aqueles com os maiores valores de aptidão, tenham uma maior probabilidade de serem submetidos ao operador de variação.

A variação possibilita a geração de novos indivíduos (soluções) dentro do espaço de busca pela perturbação daquelas existentes. Usualmente, a variação em um AE dá-se através de duas formas: a troca de informações entre pares de indivíduos (ancestrais) criando dois novos indivíduos (descendentes) e da geração de novas informações e/ou recuperação de informações perdidas durante a evolução, denominadas recombinação e mutação, respectivamente.

O operador de mutação possui uma importância fundamental nos métodos de PE e EE, no AG ele possui uma importância secundária. O operador de recombinação não é comumente utilizado na PE, mas possui importância secundária na EE e é o operador principal de variação do AG.

Segundo Bäck et al. [15], as principais diferenças entre os métodos-chaves dos AEs (AG, PE e EE) são:

- Representação dos indivíduos;
- Projeto dos operadores de variação (recombinação e/ou mutação);

- Mecanismo de seleção/reprodução.

PE e EE utilizam indivíduos diretamente representados por um par de vetores de valores reais $\mathbf{v} = (\mathbf{x}, \sigma)$, onde o primeiro vetor $\mathbf{x} \in \Re^n$ representa um ponto no espaço de busca e o segundo vetor $\sigma \in \Re^n$ indica os desvios padrões. Variantes híbridas inteira/real existem para os dois métodos. Por sua vez, AGs utilizam indivíduos que são comumente codificados como cadeias binárias, embora outras formas de representação sejam possíveis, tais como real e inteira (esta última sendo muito útil para problemas combinatórios que expressam ordenamentos, agrupamentos etc.). Programação genética (PG), uma extensão do AG, utiliza estruturas complexas de árvores como indivíduos [135]. Como programas de computador podem ser representados como árvores, a PG possibilita a evolução destes programas de forma automática. Segundo Koza [136], a PG é uma tentativa para tratar com uma das questões centrais da ciência da computação:

> *"Como os computadores podem aprender a programar a si próprios para resolverem problemas sem serem explicitamente programados?"*

3.5 COMPARAÇÃO DE ALGORITMOS: TEOREMAS *NO FREE LUNCH*

Com respeito à questão de comparação de algoritmos para resolver POCs, Wolpert e Macready [137] reportaram dois importantes teoremas, denominados em inglês *No Free Lunch* (NFL), para ambos os tipos de problemas estáticos (teorema 1) e dinâmicos (teorema 2). Em resumo, a performance média de qualquer par de algoritmos através de todos os possíveis problemas de otimização é idêntica (para ambos os casos estático e dinâmico). Isto significa em particular que, se um algoritmo a_1 é superior (na média) a um outro algoritmo a_2 para uma classe de problemas, então a_1 deve ser inferior (na média) a a_2 para o restante dos outros problemas. A definição dos autores de um algoritmo inclui todos os métodos de otimização caixa-preta (propósito geral), tais como algoritmos evolutivos e têmpera simulada.

Nenhuma comparação de algoritmos deveria ser reportada dizendo que a_1 é melhor do que a_2 sem que uma nota do tipo "nas funções testadas" ou "sob as suposições pré-escritas" seja evidenciada. Uma comparação de algoritmos sobre uma pequena amostra de problemas não é significativa. É importante ressaltar que os teoremas reportados por Wolpert e Macready assumem que os algoritmos não incluem nenhum conhecimento específico sobre os problemas a serem resolvidos. Os próprios autores concluíram que seus resul-

tados indicam a importância de incorporar algum conhecimento específico do problema abordado no comportamento do algoritmo.

Os teoremas NFL geraram muita controvérsia na comunidade de AEs [138]. Em parte, devido ao fato de que um dos algoritmos comparados (a_1 ou a_2) pode ser um algoritmo aleatório. Como Culberson [138] salientou, existe ainda hoje uma persistência por parte dos pesquisadores em descobrir um algoritmo que possa se sobressair sobre todos os restantes (i.e. o algoritmo "mágico").

3.5 CONCLUSÃO

Neste capítulo, nós caracterizamos a classe de problemas de otimização combinatória (POCs) e apresentamos heurísticas que têm sido amplamente utilizadas para resolvê-los. Especificamente, têmpera simulada e algoritmos genéticos são duas ferramentas poderosas para serem usadas com POCs, podendo, inclusive, serem utilizadas em conjunto. Por fim, nós discutimos sobre a comparação da performance destas heurísticas para resolver POCs. Não é correto dizer que uma heurística é melhor do que outra baseado apenas na observação de uma pequena amostra de problemas solucionados.

CAPÍTULO 4

ALGORITMOS GENÉTICOS

4.1 INTRODUÇÃO

O algoritmo genético (AG) é um método de otimização e busca estocástico que imita, de uma forma muito simplificada, o processo de evolução natural das espécies. Este método foi desenvolvido pelo professor John Holland (Universidade de Michigan, EUA) durante a década de 60, que estava interessado no projeto e implementação de sistemas adaptativos [123, 140]. Em 1975, Holland publicou um livro pioneiro sobre o assunto, intitulado em Inglês "*Adptation in Natural and Artificial Systems*" (University of Michigan Press), que apresentava a fundamentação teórica do seu método. Posteriormente, o trabalho de Holland foi estendido por vários alunos seus de doutorado, tais como A. D. Bethke, K. A. Jong, R. B. Hollstein e D. Goldberg.

Um AG transforma uma população de indivíduos, tipicamente cadeias de caracteres de comprimento fixo, em uma nova geração da população usando os operadores de seleção e variação (recombinação e mutação). Cada indivíduo possui um valor de aptidão associado que mede a sua qualidade em relação ao problema abordado. Quanto maior o valor de aptidão de um indivíduo, maior é a sua chance de sobreviver e de se reproduzir, e conseqüentemente, de passar parte de seu código genético para a geração seguinte.

Os operadores de recombinação e mutação manipulam as variáveis de decisão codificadas, e não diretamente as próprias variáveis de decisão do problema (exceto quando a representação de ponto flutuante é utilizada) [73]. Embora a codificação binária seja tradicional no uso de AGs para uma ampla gama de problemas [126], outras representações são possíveis e muito utilizadas também, como a representação de ponto flutuante (problemas de engenharia de controle e de projetos) e inteira (problemas de engenharia de manufatura). A utilização da representação de ponto flutuante é descrita em detalhes por [71]; a representação binária e sua utilização são abordadas por [123].

A tabela 4.1 apresenta um resumo da terminologia básica empregada pela comunidade de pesquisadores em AGs (vocabulário obtido da genética natural) [141].

Tabela 4.1 – Terminologia em AGs

Genética Natural	Algoritmo Genético (AG)
Cromossomo	Conjunto de parâmetros codificado (indivíduo)
Gene	Característica ou bit
Alelo	Valor da característica ou bit
Locus	Posição de cada bit na cadeia
Genótipo	Estrutura genética de um indivíduo
Fenótipo	Indivíduo decodificado dando uma possível solução ao problema
Geração	Um ciclo de evolução da população
População	Um conjunto de indivíduos

Os três operadores – seleção, recombinação e mutação – são utilizados iterativamente para permitir a evolução a partir de uma população inicial. Como cada indivíduo da população representa uma solução codificada do problema, este processo iterativo permite que o espaço de busca possa ser explorado paralelamente por cada indivíduo. Este processo iterativo é finalizado quando um critério de parada é satisfeito. Geralmente, um número máximo de gerações é utilizado como critério de parada. O ciclo básico de funcionamento do AG é apresentado na Figura 4.1.

Após a geração de uma população inicial P(t) (etapa de inicialização), os operadores de seleção, recombinação e mutação são aplicados seqüencialmente em P(t). O operador de seleção, ao contrário dos operadores de variação que operam diretamente com a representação genética, utiliza somente a distribuição dos valores de aptidão de P(t) [15].

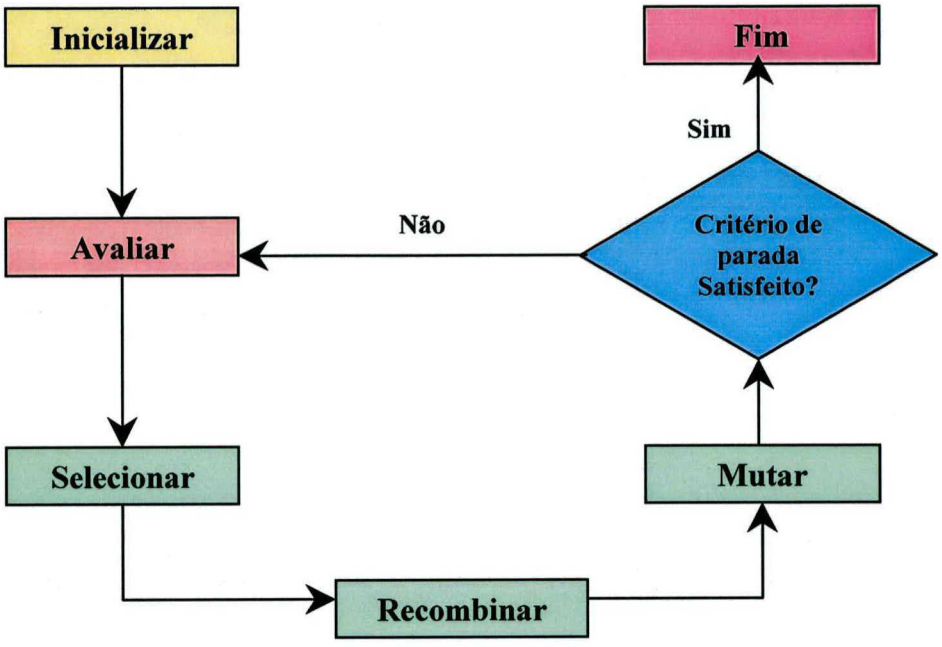


Figura 4.1 – Ciclo básico de um AG.

4.2 FUNÇÃO OBJETIVO E FUNÇÃO DE APTIDÃO

A função objetivo f(x) fornece uma medida qualitativa de como cada indivíduo se comporta para um dado problema. Se nós desejamos otimizar uma função, dois casos podem ser considerados. No primeiro, se um problema de maximização é requerido, os indivíduos mais aptos dentro de uma população P(t) serão aqueles com os maiores valores de f(x). No segundo, similarmente, se um problema de minimização é requerido, os indivíduos que fornecerem os menores valores de f(x) serão considerados os mais aptos dentro de P(t). Entretanto, dois problemas surgem quando usamos f(x) diretamente dentro de um AG básico (como a função de aptidão):

- O AG básico somente trabalha de forma a maximizar f(x);
- O AG básico utiliza um método de seleção proporcional e não pode aceitar valores negativos retornados por f(x).

Desta forma, um mapeamento do tipo $\Phi(x) = \Theta(f(x))$ é necessário quando se deseja minimizar f(x), dando aos indivíduos mais aptos os valores de $\Phi(x)$ mais altos, ou quando f(x) retorna valores negativos. Assim, a função de aptidão realmente utilizada dentro de um AG é $\Phi(x)$, e é a através dos valores obtidos a partir desta função que os indivíduos serão selecionados. O conjunto de valores de aptidão de uma população é denominada de distribuição dos valores de aptidão.

Existem várias formas de se implementar o mapeamento da função f(x) para $\Phi(x)$, a qual podemos citar o escalonamento linear, truncamento sigma e ranking [73, 123, 142]. Estas formas de mapeamento são também utilizadas para exercer um certo controle na pressão seletiva. A atribuição dos valores de aptidão baseada em ranking é utilizada neste trabalho e, portanto, será descrita em maiores detalhes a seguir.

4.2.1 ATRIBUIÇAO DOS VALORES DE APTIDÃO BASEADA NO RANKING

Inicialmente, os indivíduos de P(t) são ordenados de acordo com seus valores da função objetivo, sendo que ao pior indivíduo é atribuída a 1º posição do ranking e ao melhor indivíduo a última posição (ordenamento decrescente). O valor de aptidão de um indivíduo depende somente da sua posição no ranking e não do valor da função objetivo que ele possui [143]. A distribuição dos valores pode variar de forma linear ou não-linear com o ranking .

A atribuição do valor de aptidão para um indivíduo baseado em um ranking linear é dado por

$$\Phi(x) = (2 - PS) + 2 \cdot (PS - 1) \cdot \left[{}^{(i-1)}\!/_{N-1} \right], \tag{4.1}$$

onde

PS = Pressão seletiva, PS $\in$ [1,2];

N = Tamanho da população P(t);

i = Posição do indivíduo no ranking.

Como exemplo, considere a população P(t) e o correspondente conjunto de valores da função objetivo f(x) dado na tabela 4.2. A obtenção da distribuição dos valores de aptidão de P(t) é mostrada nas duas últimas colunas da tabela 4.2 (ela apresenta os valores de $\Phi(x)$ para PS igual a 1.1 e 2.0). O método de ranking permite um controle direto sobre a pressão seletiva. Repare que todos indivíduos possuem um valor $\Phi(x)$ diferente, inclusive aqueles que possuem o mesmo valor de f(x) (indivíduos 1 e 9), pois o valor de $\Phi(x)$ é baseado na posição. Os valores de aptidão dos indivíduos 1 e 9 são ressaltados por um asterisco na tabela 4.2.

Tabela 4.2 – Atribuição dos valores de aptidão
baseado no ranking dos indivíduos

Indivíduo	Valor de f(x)	Indivíduos ordenados	Valor de f(x) ordenado	Φ(x) (PS = 1.1)	Φ(x) (PS = 2.0)
1	10.2	5	23.7	0.9000	0
2	4.23	6	16.9	0.9222	0.2222
3	1.00	1*	10.2*	0.9444*	0.4444*
4	3.05	9*	10.2*	0.9667*	0.6667*
5	23.7	7	9.11	0.9889	0.8889
6	16.9	8	5.67	1.0111	1.1111
7	9.11	2	4.23	1.0333	1.3333
8	5.67	10	3.56	1.0556	1.5556
9	10.2	4	3.05	1.0778	1.7778
10	3.56	3	1.00	1.1000	2.0000

4.3 OPERADOR DE SELEÇÃO

Este operador emprega o princípio de sobrevivência dos indivíduos mais aptos dentro de uma população: estes indivíduos possuem uma maior probabilidade de contribuir com uma ou mais cópias de si mesmos para o processo de recombinação do código genético, do que aqueles que são menos aptos (que têm uma baixa probabilidade de serem escolhidos). Supõe-se que a recombinação de indivíduos mais aptos gerem descendentes melhores (soluções de melhor qualidade).

Usualmente empregam dois tipos de substituição da população P(t): geracional e estacionária. O primeiro esquema de substituição considera que a cada geração a população de ancestrais P(t) é totalmente substituída pela nova população P(t+1) (formada pelos descendentes). Desta forma, é possível que o melhor indivíduo da geração corrente t venha a ser substituído por um outro menos apto na geração t+1 (isto pode ocorrer se o melhor indivíduo for destruído pelo operador de recombinação e/ou mutação). Para prevenir este tipo de situação, pode-se empregar a estratégia elitista. Esta estratégia consiste simplesmente em inserir o melhor indivíduo de P(t) em P(t+1) [128].

Para grandes populações, o número de indivíduos de P(t) inseridos em P(t+1) pode ser maior (tipicamente até 10% são inseridos).

Na substituição estacionária, em vez de toda a população P(t) ser substituída de uma única vez, somente alguns poucos indivíduos (tipicamente um ou dois) são selecionados e submetidos aos operadores de variação, sendo então inseridos na população P(t+1) [82, 144].

4.3.1 SELEÇÃO POR TORNEIO

O método de seleção por torneio funciona da seguinte forma: q indivíduos são escolhidos aleatoriamente a partir da população P(t) e aquele com o maior valor de aptidão dentre estes q indivíduos é considerado o vencedor. Cada vencedor é colocado na população Q(t) (a ser submetida ao operador de recombinação). Este processo é repetido λ vezes, até que o tamanho da população Q(t) seja atingido.

Usualmente, os torneios são realizados entre dois indivíduos (torneio binário, $q = 2$), mas torneios com um grupo de tamanho $q > 2$ também podem ser utilizados. O parâmetro q é denominado de tamanho do torneio. Algumas características interessantes deste método são:

- Escolha determinística do indivíduo;
- Permite um ajuste da pressão seletiva através do parâmetro q (q é proporcional à pressão seletiva) [15, 145];
- Pode ser facilmente estendido a problemas envolvendo múltiplos objetivos [84];
- Pode-se garantir múltiplas cópias do indivíduo mais apto em Q(t) [141];
- Possui complexidade de tempo igual a O(N), onde N é o tamanho de P(t) (conforme o algoritmo da Figura 4.2) [150].

O algoritmo do método de seleção por torneio é apresentado na Figura 4.2 [146].

```
Algoritmo da Seleção por Torneio
Entrada:
        P(t) (população);
        q ∈ {1, 2,..., N} (tamanho do torneio);
        Φ(t) (distribuição dos valores de aptidão de P(t));
Saída:
        Q(t) (população obtida após o processo de seleção);
Para i ← 1:N
        q ← pegar_q_indivíduos(P(t)); % processo aleatório
        Q(t) ← indivíduo_mais_apto(q,Φ(t));
fim_para
retorne {Q(t)}
```

Figura 4.2 – Pseudocódigo da seleção por torneio.

4.3.2 SELEÇÃO POR TRUNCAMENTO

Este método de seleção é utilizado pelo AG desenvolvido por [147]. Somente os T% melhores indivíduos de P(t) são selecionados para Q(t). O parâmetro T, denominado coeficiente de truncamento, é normalmente escolhido na faixa de 50% a 10%.

O método de seleção por truncamento é equivalente a seleção (μ,λ) utilizada pela estratégia evolutiva, onde μ ancestrais produzem λ descendentes ($\lambda > \mu$) e os melhores μ descendentes são selecionados para substituir os ancestrais, com $T = 100*\mu/\lambda$ [148].

O algoritmo do método de seleção por truncamento é apresentado na Figura 4.3 [146]. Desde que um ordenamento é requerido, este método tem uma complexidade de tempo igual a O(N*lnN).

Algoritmo da Seleção por Truncamento

Entrada:

P(t) (população);

$T \in [0,1]$ (coeficiente de truncamento);

$\Phi(t)$ (distribuição dos valores de aptidão de P(t));

Saída:

Q(t) (população obtida após o processo de seleção);

% ordenar P(t) de acordo com os valores de aptidão de forma ascendente

$P^*(t) \leftarrow$ ordenar_P($\Phi(t)$);

Para $i \leftarrow 1:N$

$r \leftarrow$ aleatório{[(1-T)*N],...,N};

$Q(t) \leftarrow P_r^*(t)$;

fim_para

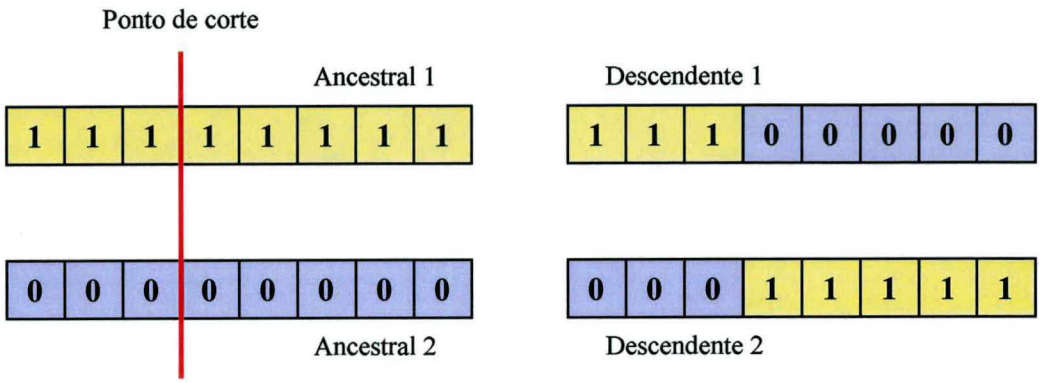
retorne {Q(t)}

Figura 4.3 – Pseudocódigo da seleção por truncamento.

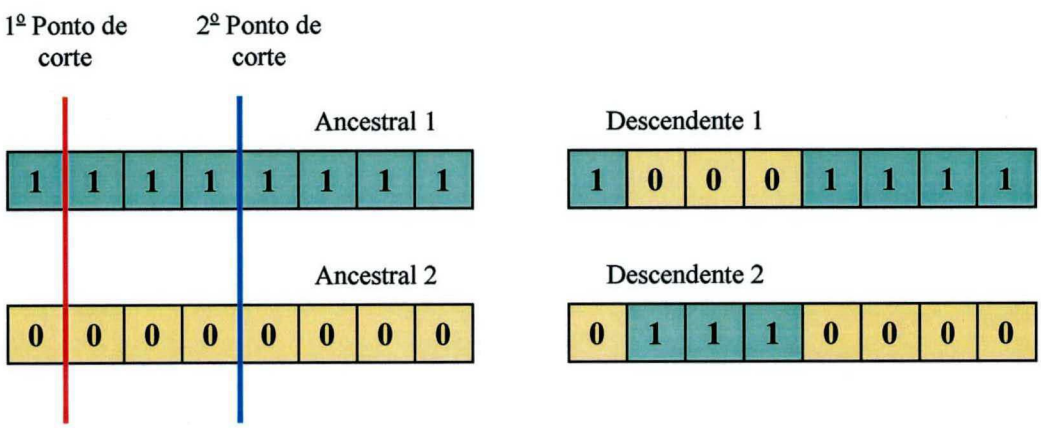
4.4 OPERADOR DE RECOMBINAÇÃO

O operador de recombinação é utilizado no AG para trocar o material genético entre pares de indivíduos. Esta recombinação ocorre com probabilidade fixa p_r (denominada de probabilidade de recombinação), mas um ajuste adaptativo deste parâmetro, sem a intervenção do usuário, também é possível [149].

O operador de recombinação mais simples é o de 1 ponto: dado um par de ancestrais de comprimento k, um ponto de corte (um número inteiro) é escolhido aleatoriamente no intervalo [1, k) para ambos os indivíduos. O material genético à direita do ponto de corte de cada ancestral é trocado pelo o do seu parceiro, gerando dois novos descendentes (Figura 4.4a). Quando dois pontos de corte são utilizados (recombinação de 2 pontos), os materiais genéticos nos extremos são deixados nos ancestrais e somente a parte do meio é trocada entre eles (Figura 4.4b). Obviamente, os dois pontos de corte são diferentes.



(a) Recombinação com um ponto de corte.



(b) Recombinação com dois pontos de corte.

Figura 4.4 – Operadores de recombinação

4.5 OPERADOR DE MUTAÇÃO

O operador de mutação é tradicionalmente visto como um operador de variação secundário dentro do AG, entretanto, ele tem um papel fundamental no funcionamento do AG. Este operador é responsável por introduzir diversidade na população, explorando novas regiões no espaço de busca ou mesmo restaurando material genético inadvertidamente perdido durante o processo de busca.

Utilizando uma representação binária, cada bit na cadeia de um indivíduo é invertido (se é 1 então vira 0 ou vice-versa) com uma dada probabilidade fixa P_m. Em uma população de tamanho N cujo comprimento dos indivíduos é dado por L, aproximadamente P_m*M*L mutações ocorrem por geração.

Um ajuste automático de P_m também é descrito em [146]. Tipicamente, a faixa de valores para a probabilidade de mutação é dada por [0.001,0.01] .

O funcionamento do tipo de operador mais simples é apresentado na Figura 4.5, mas existem outros tipos de operadores que foram propostos para trabalhar com outras representações, tais como a real [71].

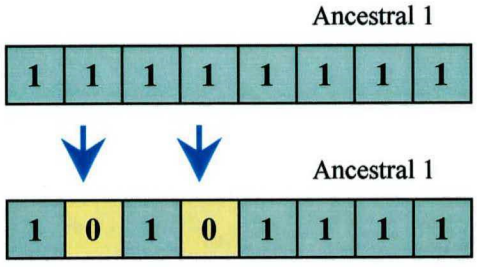


Figura 4.6 – Operação de mutação.

4.6 BALANÇO ENTRE EXPLORAÇÃO (EXPLORATION) E EXPLORAÇÃO (EXPLOITATION)

A eficiência do AG para solucionar problemas depende de um balanço entre duas propriedades: exploração (*exploration*) e exploração (*explotation*). A primeira propriedade refere-se a ação de explorar novas e desconhecidas regiões do espaço de busca, enquanto que a segunda refere-se a ação de explorar informações obtidas a partir de pontos já visitados para encontrar pontos ainda melhores [142, 144]. Estas duas propriedades estão inversamente relacionadas: a primeira aumenta a diversidade da população (menor pressão seletiva), enquanto que a segunda reduz esta diversidade em detrimento da busca ser focalizada nos indivíduos mais aptos da população (maior pressão seletiva) [82]. Portanto, um balanço entre exploração (*exploration*) e exploração (*explotation*) é crítico para o sucesso do AG. Os seguintes parâmetros permitem afetar este balanço [82, 144, 150];

- Probabilidade de recombinação (P_r): Um valor alto de P_r tende a tornar a busca demasiadamente localizada.
- Probabilidade de mutação (P_m): Um alto valor de P_m tende a tornar o AG um método de busca aleatório. A diversidade introduzida na população é controlada pelo valor de P_m;
- Tamanho da população (N): O aumento de N tende a aumentar a diversidade de P(t), entretanto, a custa de um aumento proporcional no tempo de execução do AG.

Pequenas populações possuem maior probabilidade de convergirem prematuramente para um valor ótimo local;

- Pressão seletiva (PS): É diretamente controlada pelo usuário quando a atribuição dos valores de aptidão é baseada no ranking (linear ou não-linear) dos indivíduos.

4.7 ADAPTAÇÃO DE AGs PARA TRATAR PROBLEMAS COM RESTRIÇÕES

A grande maioria dos problemas de otimização na engenharia possui restrições que devem ser satisfeitas. A presença destas restrições afeta de forma significativa a performance de qualquer método de otimização, incluindo algoritmos evolutivos [158]. Como Blickle [128] salientou, não existe até o momento nenhuma abordagem universal para manipular restrições dentro de AEs, contudo, algumas abordagens têm sido amplamente empregadas, tais como a penalização ou o reparo de indivíduos não-factíveis. As referências [151, 152, 153, 154] fornecem uma excelente revisão de muitas abordagens para a manipulação de restrições que foram propostas para AEs. As principais abordagens propostas até aqui são resumidas a seguir.

1. Uso de funções de penalização: Esta é a abordagem mais comum na literatura sobre AEs e, principalmente, AGs [151, 154]. O domínio da função objetivo (f) é extendido para a seguinte forma:

(4.2) $g_i(x) = f_i(x) \pm Q_i$,

onde Q_i representa ou uma penalidade para um indivíduo i ou um custo para tornar o indivíduo i factível. Obviamente, se o indivíduo i é factível, então Q_i é igual a zero. O termo Q_i geralmente incorpora as restrições de igualdade e/ou inequações que, quando violadas, adicionam um alto custo à avaliação de um indivíduo não-factível i. Richardson et al.[155] formulou a hipótese de que as penalidades que são funções da distância a partir da região de factibilidade são melhores do que aquelas que são meramente funções do número de restrições violadas.

2. Operadores de reparo: Em alguns problemas de otimização combinatória, tais como o PCV, problema da mochila etc., pode ser relativamente fácil reparar um indivíduo não-factível, isto é, torná-lo factível [152]. Entretanto, o operador de reparo é dependente do problema a ser resolvido e, desta forma, não há regras gerais para implementar tal opera-

dor. E, muito provavelmente, para alguns problemas a reparação da não-factibilidade de um indivíduo pode ser tão difícil quanto resolver o problema original [128, 152].

Segundo Coit et al. [156], em alguns casos a tarefa de encontrar uma solução factível é, ela própria, um problema NP-difícil (pertence a classe de problemas NP-completos). O indivíduo reparado, ou substitui o indivíduo original em Q(t) com alguma probabilidade, ou ele é usado para avaliação da função objetivo somente.

3. Rejeição de indivíduos não-factíveis: Consiste em eliminar os indivíduos não-factíveis de uma população Q(t). É o método mais fácil (trivial) de tratar com a manipulação de restrições em AEs. Entretanto, quando o problema é altamente restringido, ele é muito ineficiente para AGs (a geração de um indivíduo factível para substituir um não-factível pode consumir excessivo tempo computacional). Este método é usado na estratégia evolutiva (EE).

4.8 ALGORITMOS GENÉTICOS PARALELOS E DISTRIBUÍDOS

A implementação paralela de um AG possibilita uma maior velocidade de execução dos AGs e, dependendo da forma como isto é feito, melhores resultados são obtidos quando comparados com um AG básico (implementado seqüencialmente) [157].

Existem basicamente três classes de AGs paralelos: AGs de fina granularidade (*fine-grained parallel GAs*), AGs de grossa granularidade (*coarse-grained parallel GAs*) e AGs do tipo mestre-escravos [158]. Na primeira classe, indivíduos são distribuídos em uma malha de processadores, onde cada processador é responsável por um indivíduo (ver Figura 4.7) [159]. Tipicamente, os indivíduos da população são colocados em uma grade planar visto que muitos computadores maciçamente paralelos possuem processadores conectados com esta topologia. Entretanto, outras topologias podem ser simuladas também [158]. Schwehm fez um estudo sobre o uso de diferentes topologias, a saber: anel, toroidal, cubo 16x8x8, hipercubo 4x4x4x4x4 e um hipercubo binário 10-D [158].

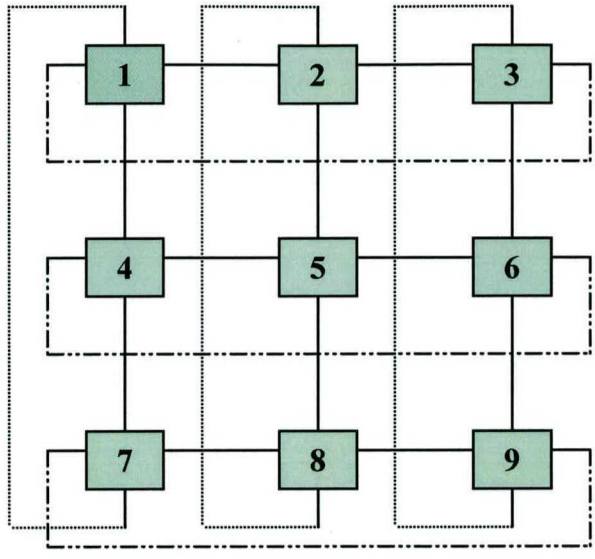


Figura 4.7 – AG de fina granulometria: cada processador executa um indivíduo (aqui, cada indivíduo tem exatamente 4 vizinhos).

A seleção e recombinação dos indivíduos são feitas localmente, de acordo com a vizinhança de cada indivíduo (a dimensão da vizinhança deve ser pequena). As vizinhanças são parcialmente coincidentes. Isto é, a intersecção das vizinhanças pode, dependendo da localização dos indivíduos escolhidos e do tamanho da dimensão da vizinhança, não ser um conjunto vazio. Como conseqüência deste fato, boas soluções podem ser disseminadas por toda a população [158, 160]. O AG desenvolvido por Mühlenbein [147, 161] permite que cada indivíduo realize um procedimento de busca local *hill-climbing* em sua vizinhança, melhorando seu valor de aptidão durante todo o seu tempo de vida dentro da população.

Na segunda classe, k diferentes AGs seqüenciais rodam em paralelo, cada AG sendo atribuído a um processador. Os AGs paralelos distribuídos trabalham da seguinte forma. Múltiplas populações evoluem isoladamente uma da outra, sendo que cada população é atribuída a um processador que executa um AG seqüencial. (Figura 4.8).

Periodicamente, ocorre uma troca dos melhores indivíduos (migração) entre as populações vizinhas. A vizinhança de uma população depende da topologia de interconexão dos processadores, tipicamente uma topologia hipercubo. Para realizar o processo de migração, dois parâmetros devem ser definidos [158]:

- Intervalo de migração: especifica o número de gerações a ser transcorridas entre cada processo de migração;

- Taxa de migração: especifica a porcentagem de indivíduos selecionados para migrar (os melhores da população).

O funcionamento desta classe de AGs paralelos é assíncrono, apesar do processo de migração ser síncrono (ocorre em intervalos constantes pré-determinados para todas as populações). O processo de migração pode também ocorrer de forma assíncrona [158], isto é, com a ocorrência de algum evento discreto no tempo. Alguns modelos assumem que as populações evoluem até que cada uma converge para uma única solução e, neste ponto, ocorre uma troca de um número arbitrário de indivíduos com o propósito de restaurar a diversidade dentro das populações e prevenir a população de uma convergência prematura para indivíduos de baixa qualidade [158].

O processo de migração permite manter um conhecimento global do espaço de busca, em vez de confinar localmente as informações obtidas por cada população [159].

```
Processador k
t ← 1;
criar P(t);
avaliar P(t);
enquanto {critério de parada = falso}
        selecionar Q(t)  a partir de P(t);
        recombinar Q(t);
        mutar Q(t);
        avaliar Q(t);
        se {tempo de migrar == verdadeiro}
            enviar q(t) para os vizinhos;
            receber q(t) dos vizinhos;
            inserir q(t) dentro de Q(t);
        fim_se
        P(t) ← Q(t);
        t ← t + 1;
fim_enquanto
retorne {Q(t)}
```

Figura 4.8 - Pseudocódigo do AG paralelo distribuído.

Assim como os AGs pertencentes à segunda classe, os AGs da terceira classe também rodam sequencialmente em k diferentes processadores. Entretanto, existe uma diferença fundamental aqui: eles são centralizados por meio de um processador mestre que efetua o processo de seleção global e não distribuído. A implementação centralizada tem uma estrutura mestre-escravo (Figura 4.9).

O processador mestre realiza a seleção dos indivíduos baseado na distribuição dos valores de aptidão da população inteira e distribui uma fração dos indivíduos selecionados para cada processador escravo aplicar os operadores de recombinação e mutação nestes indivíduos. Após feito isto, os processadores escravos avaliam suas populações e as retornam juntamente com seus respectivos valores de aptidão para o processador mestre realizar uma nova seleção. Esta interação entre processadores escravos e mestre se repete até que algum critério de parada seja satisfeito. O problema desta implementação é que o processador mestre somente pode realizar o processo de seleção após o término de execução de todos os processadores escravos (esta implementação trabalha de forma síncrona). Implementações assíncronas também são possíveis. O AG com estrutura mestre-escravo síncrono tem as mesmas propriedades de um AG seqüêncial, exceto que a velocidade de execução é diferente [158].

Processador Mestre	Processador Escravo #1

t ← 1;
criar P(t);
avaliar P(t);
enquanto {critério de parada = falso}
selecionar Q(t) a partir de P(t);
enviar Q(t) para os k escravos;
receber Q(t) e Φ(t) dos k escravos;
P(t) ← Q(t);
t ← t + 1;
fim_enquanto
retorne {Q(t)}
	receber Q(t) do mestre;
	recombinar Q(t);
	Mutar Q(t);
	Avaliar Q(t);
	retorne {Q(t), Φ(t) }

	Processador Escravo #k
	receber Q(t) do mestre;
	recombinar Q(t);
	Mutar Q(t);
	Avaliar Q(t);
	retorne {Q(t), Φ(t) }

Figura 4.9 – Pseudocódigo do AG paralelo centralizado.

4.9 CONCLUSÃO

Este capítulo procurou aprofundar alguns conceitos sobre os algoritmos genéticos, especificamente caracterizando suas partes fundamentais e seu funcionamento básico. Tópicos avançados como o tratamento de restrições e a paralelização de AGs também foram sucintamente apresentados.

CAPÍTULO 5

UMA ABORDAGEM EVOLUTIVA APLICADA AO PROBLEMA DE FORMAÇÃO DE CÉLULAS DE MANUFATURA

5.1 INTRODUÇÃO

Neste capítulo, o problema de formação de células de manufatura (PFCM) é formalmente definido e dois modelos matemáticos são apresentados para particionar conjuntos de máquinas e peças de um sistema de manufatura. A dimensão do espaço de busca também é formalmente descrita. Os modelos consideram aspectos importantes e reais de um ambiente de manufatura, tais como a demanda de produção, seqüência de operações das peças, restrições no número de máquinas por células, rotas de processamentos alternativos e a carga de trabalho em cada máquina. Entretanto, os dois modelos não são resolvidos simultaneamente.

5.2 DEFINIÇÃO DO PROBLEMA

Nós definimos $\widetilde{M} = \{m_1, m_2, ..., m_M\}$ e $\widetilde{N} = \{n_1, n_2, ..., n_N\}$ como sendo os conjuntos formados por M máquinas e N tipos de peças, respectivamente. O problema de formação de células consiste em encontrar as partições $\widetilde{C} = \{C_1, C_2, ..., C_L\}$ e $\widetilde{F} = \{F_1, F_2, ..., F_L\}$ tal que $C_s \cap C_t = \varnothing$ e $F_s \cap F_t = \varnothing$ para $s \neq t$ e $\{s,t\} \in \{1,2,..., L\}$, $\bigcup_{s=1}^{L} C_s = \widetilde{M}$ e $\bigcup_{s=1}^{L} F_s = N$ [100].

Os elementos das partições $\widetilde{C}$ e $\widetilde{F}$ representam as células de máquinas e família de peças, respectivamente. O particionamento do sistema de manufatura em um arranjo celular pode, eventualmente, ter que satisfazer um vetor de restrições de projeto **G**. A não violação das restrições de projeto assegura um particionamento factível do sistema de manufatura.

Nós definimos os conjuntos de partições factíveis para $\widetilde{C}$ e $\widetilde{F}$ como sendo $\widetilde{C}_f \subseteq \widetilde{C}$ e $\widetilde{F}_f \subseteq \widetilde{F}$, respectivamente. Para obter partições $\widetilde{C}_f$ e $\widetilde{F}_f$ que minimizam o custo total de projeto, um vetor de objetivos **F** deve ser minimizado. Nós podemos apresentar agora uma definição formal para o problema de formação de células.

Definição 5.1 (Formação de Células de Manufatura): *Encontrar* $\widetilde{C}^* \in \widetilde{C}_f \wedge \widetilde{F}^* \in \widetilde{F}_f \mid$ *min{**F**}.*

5.2.1 TAMANHO DO ESPAÇO DE BUSCA DO PROBLEMA

Os modelos matemáticos apresentados neste capítulo, e, portanto, utilizados neste trabalho, consideram que o número de células deve ser definido a *priori*. O número de caminhos para particionar um conjunto de M objetos (máquinas) dentro de L subconjuntos não vazios (células), é dado pelo número de Stirling de segundo tipo [9, 162]:

$$\left\{ \begin{matrix} M \\ L \end{matrix} \right\} = \frac{\left[\sum_{k=1}^{L} \binom{L}{k} \cdot k^{M} \cdot (-1)^{L-k} \right]}{L!} \tag{5.1}$$

onde

$\binom{L}{k} = L!/[k! \cdot (L-k)!]$ é o coeficiente binomial que fornece o número de caminhos para escolher um subconjunto de k elementos a partir de um conjunto de L elementos (inteiros $L \geq k \geq 0$).

A seguinte fórmula de recorrência pode ser utilizada para calcular o número de Stirling de segundo tipo:

$$\left\{ \begin{matrix} M \\ L \end{matrix} \right\} = \left\{ \begin{matrix} M-1 \\ L-1 \end{matrix} \right\} + L \cdot \left\{ \begin{matrix} M-1 \\ L \end{matrix} \right\} \tag{5.2}$$

com

$$\left\{ \begin{matrix} M \\ M \end{matrix} \right\} = 1 \text{ e } \left\{ \begin{matrix} M \\ 1 \end{matrix} \right\} = 1.$$

Por exemplo, existem 7 caminhos distintos para particionar um conjunto $\tilde{M} = \{1,2, 3,4\}$ (4 tipos de máquinas) em duas células:

$$\left\{ \begin{matrix} 4 \\ 2 \end{matrix} \right\} = \frac{\left[\sum_{k=1}^{2} \binom{2}{k} \cdot k^4 \cdot (-1)^{4-k} \right]}{2!} = \frac{\binom{2}{1} \cdot 1^4 \cdot (-1)^3 + \binom{2}{2} \cdot 2^4 \cdot (-1)^2}{4} = 7.$$

Nós podemos manualmente escrever estas partições: $\{1,2,3\} \cup \{4\}$; $\{1,2\} \cup \{3,4\}$; $\{1\} \cup \{2,3,4\}$; $\{1,3\} \cup \{2,4]\}$; $\{1,4\} \cup \{2,3\}$; $\{1,3,4\} \cup \{2\}$; $\{1,2,4\} \cup \{3\}$.

Entretanto, se há um conjunto de 12 máquinas que devem ser agrupadas em 3 células, o número de partições rapidamente cresce para 86.526. Se o número de tipos de máquinas é dobrado, deixando igual o número de células, existem 4.706.300.806 partições distintas (tabela 5.1). A medida que a dimensão deste problema aumenta, i.e., o número de máquinas e/ou células, o número de partições distintas rapidamente torna-se imenso.

Tabela 5.1 – Números de partições para algumas configurações (M/L)

M (máquinas)	L (células)	Partições possíveis (ordem)
12	3	82.526 (10^4)
24	3	47.063.200.806 (10^{10})
30	3	34.314.651.811.700 (10^{13})
12	4	14.676.024 (10^7)
24	4	11.681.056.634.500 (10^{13})
30	4	48.004.081.105.000.000 (10^{16})

O número total de caminhos para particionar M objetos (máquinas) dentro de subconjuntos (células) é dado pelo número de Bell [162], i.e.,

$$B_M = \sum_{k=1}^{M} \left\{ \begin{matrix} M \\ L \end{matrix} \right\}. \qquad (5.3)$$

Este número é a soma dos números de Stirling de segundo tipo. Por exemplo, existem 5 caminhos para particionar 3 máquinas em células: {1,2,3}; {1,2}∪{3}; {1,3}∪{2}; {1}∪{2,3}; {1}∪{2}∪{3}.

O número de células mínimo é 1, sendo todas as máquinas agrupadas dentro dela, e o máximo é M, onde cada máquina i é atribuída à sua célula particular. Desta forma, o número total de caminhos na qual as atribuições máquina-célula podem ser feitas é dado por [174]:

$$\sum_{L=1}^{M} \left\{ {M \atop L} \right\} = \sum_{L=1}^{M} \left\{ \left[\sum_{k=1}^{L} \binom{L}{k} \cdot k^{M} \cdot (-1)^{L-k} \right] \Big/ L! \right\}. \qquad (5.4)$$

Quando é considerado que cada peça j possui um conjunto $\widetilde{R} = \{r_1, r_2, ..., r_p\}$ de rotas de processamentos, na qual uma deve ser escolhida para a efetiva manufatura da peça j, o número de configurações possíveis de rotas é dado por:

$$CR = \prod_{j=1}^{N} p_j , \qquad (5.5)$$

onde $p_j = |\widetilde{R}|$ (p é um número inteiro positivo).

Um caso especial ocorre quando todas as peças possuem o mesmo número de rotas, i.e., $|\widetilde{R}_j| = p$ (j =1, 2, ..., N) e p > 1. Neste caso, o número de configurações é:

$$CR = p^N. \qquad (5.6)$$

Por exemplo, se há 12 peças em uma fábrica, sendo que 7 peças possuem 2 rotas de processamento, 3 peças possuem 3 rotas e 2 peças possuem 4 rotas, então pela equação (1.6) existem 55.296 configurações possíveis de rotas que podem ser escolhidas de acordo com algum critério. Entretanto, se todas as peças possuem 2 rotas de processamento, então há somente 4.096 configurações possíveis de acordo com a equação (1.7). A busca por um particionamento ótimo de um sistema de manufatura em células, i.e. encontrar $\widetilde{C}^*$ e $\widetilde{F}^*$, é muito mais complexa quando as peças possuem rotas alternativas.

Considere, por exemplo, uma fábrica que possui 12 máquinas ($\widetilde{M}$) que devem ser particionadas em 3 células e um conjunto de 12 peças ($\widetilde{N}$) que devem ser manufaturadas nestas máquinas. Considere ainda que cada peça tem 2 rotas de processamento e o critério utilizado para avaliar $\widetilde{C}$ consiste em minimizar a movimentação intercelular do chão-de-fábrica. A efetiva tarefa para obter $\widetilde{C}^*$ consiste em explorar as 82.526 partições possíveis de $\widetilde{M}$ em 3 células para cada uma das 4.096 configurações de rotas possíveis. Portanto, nós devemos dispor de uma adequada "ferramenta" computacional (*software* + *hardware*) que permita avaliar este imenso, mas finito, espaço de busca. Enumerar todas as soluções possíveis e guardar aquela com a menor movimentação intercelular é uma alternativa inviável. Entretanto, se nós relaxarmos nosso objetivo fundamental, i.e. obter $\widetilde{C}^*$, nós podemos utilizar heurísticas para avaliar uma porção deste espaço de busca para obter uma solução de boa qualidade ($\widetilde{C}_h^*$) a um custo computacional aceitável. No atual contexto, obter um agrupamento de máquinas ótimo, tal que as peças exijam uma mínima movimentação intercelular, depende justamente da configuração de rotas escolhidas para estas peças. Trata-se, portanto, de um complexo problema combinatório composto.

A seguir, nós propomos um roteiro para resolver o PFCM em três passos: (1) particionamento de um conjunto de tipos de máquinas; (2) distribuição das máquinas idênticas entre as células (não implementado neste trabalho) e (3) formação de famílias de peças.

5.2.2 ASPECTOS DA MANUFATURA

A seguir, nós listamos alguns importantes aspectos que deveriam ser levados em consideração na formação de células de manufatura, além da informação contida na matriz máquina-peça binária. Os itens 1,2 e 4 são considerados neste trabalho.

1. Seqüência de operações: A seqüência de operações pode ter um impacto significativo no custo de movimentação de materiais dentro de um sistema [64, 96, 99]. Um tipo de peça que requer uma operação intermediária fora de sua célula envolve duas transferências intercelulares: o 1º deslocamento de sua célula para uma outra e o 2º deslocamento desta célula para a sua célula de origem. Se a primeira ou última operação é feita em uma outra célula, somente uma transferência intercelular é necessária. Uma importante consideração, como enfatizado por Harhalakis et al. [96], é que a seqüência de operações de um tipo de peça poderia incluir, se fossem necessárias, operações não-consecutivas na mesma máquina. Isto é, um tipo de peça pode visitar uma máquina mais de uma vez.

2. Demanda de produção: O custo de movimentação de materiais é proporcional à demanda de produção dos tipos de peças e, por isso, é muito importante considerar este aspecto [64].

Muitas vezes a demanda de produção é considerada conhecida e constante sob um determinado horizonte de produção. Entretanto, é muito provável que esta demanda varie com o tempo, de acordo com as necessidades dos consumidores. Assim, como sugerido em [2, 163], a informação da demanda de produção poderia ser considerada imprecisa na formação de células. A demanda de produção neste trabalho é levada em consideração, entretanto, ela é conhecida e constante.

3. Restrições nas capacidades das máquinas: É importante assegurar que as capacidades de processamento das máquinas são suficientes para atender a demanda de produção de todos os tipos de peças do sistema. Se não for, é necessário duplicar as máquinas gargalos (envolvendo custos de aquisição etc.) ou então elaborar planos de processos alternativos para as peças que utilizam as máquinas gargalos.

4. Planos de processos alternativos: Na literatura, duas abordagens que consideram a manufatura de diversas formas possíveis para o mesmo tipo de peça têm-se destacado [106]:

- Consideração de que cada operação de um tipo de peça pode ser realizada por um conjunto de máquinas.
- Consideração de que cada tipo de peça tem um conjunto de diversas rotas possíveis

Neste trabalho, utiliza-se a última abordagem.

A utilização de rotas alternativas de processamento para os diversos tipos de peças ajuda a atingir células de manufatura com mínima movimentação intercelular, além é claro, de possibilitar uma maior flexibilidade no processo de formação de células de manufatura.

Cada tipo de peça j pode ter mais de uma rota de processamento, a qual define a seqüência de operações para se obter este tipo de peça totalmente acabado. Entretanto, somente uma rota pode ser atribuída para o tipo de peça j.

Nós definimos $r_{jp} = [m_g\ m_h \ldots m_i]$ como sendo a seqüência de máquinas a serem visitadas pelo tipo de peça j segundo a rota p, onde $\{m_g, m_h, m_i\} \in \tilde{M}$. O conjunto $\tilde{R}_j = \{r_1, r_2, \ldots, r_p\}$ contém todas as rotas definidas para o tipo de peça j.

Quanto maior o número de aspectos que são incluídos como dados de manufatura de entrada para o método de formação de células, mais preciso é o particionamento do sistema de manufatura em células. Entretanto, como este problema é reconhecidamente NP-completo, a inclusão de muitos destes aspectos pode até inviabilizar a utilização do método em problemas reais de grandes dimensões. Modelos matemáticos geralmente são desenvolvidos para incluir muitos aspectos, entretanto, eles somente podem ser resolvidos por métodos heurísticos (métodos enumerativos somente são utilizados para resolver problemas pequenos). Claro, a qualidade da solução irá depender da heurística empregada.

5.3 ROTEIRO PARA RESOLVER O PFCM: UMA PROPOSTA

Agora, nós iremos apresentar um roteiro para resolver o PFCM, a fim de obter uma boa solução em um tempo de computação aceitável, que é subdividido em três passos.

1. Particionamento do conjunto de tipos de máquinas: Nós consideramos somente os tipo de máquinas para criar as células de máquinas. Dada uma distribuição da demanda das peças a serem manufaturadas, nós podemos utilizar a informação da carga de trabalho induzida por todas as peças que requerem processamento na máquina i para determinar quantas unidades deste tipo de máquina devem ser adquiridas. Isto permite evitar a possibilidade de violação da capacidade da máquina i para uma dada distribuição da demanda de produção. Dados de entrada como o tempo de preparo (por lote) e o tempo de processamento são utilizados para quantificar a carga de trabalho exigida para cada máquina i. Nós salientamos que as máquinas duplicadas não são consideradas nesta etapa da resolução do PFCM. Neste trabalho, nós utilizamos um algoritmo genético para particionar um conjunto de tipos de máquinas.

2. Distribuição das máquinas duplicadas: Nesta etapa, cada peça tem sua rota de processamento analisada para se criar uma árvore de possibilidades, na qual cada ramo em profundidade a partir do nodo inicial até o nodo final da árvore especifica uma seqüência possível de processamento. Por exemplo, considere que uma peça j tenha a seguinte rota dada por $r_j = [2\ 3]$, onde cada elemento desta lista representa um tipo de máquina na qual a peça j deve ser manufaturada. Considere ainda que cada tipo de máquina possui duas unidades (máquinas 2a, 2b, 3a e 3b). Então, uma árvore de possibilidades na qual especificam as seqüências de processamento possíveis para a peça j pode ser criada (ver Figura 5.1). A escolha da rota adequada irá depender de algum determinado critério e que não ocorra nenhuma violação das restrições impostas ao PFCM.

3. Formação das famílias de peças: Uma vez que todas as máquinas sejam atribuídas às células de máquinas, as famílias de peças podem ser formadas. Nós propomos duas formas de fazer isto: uma heurística que atribui cada peça para a célula de máquina na qual ela possui o maior número de operações (ou que induz a maior carga de trabalho) ou usar uma técnica de inteligência computacional, e.g. ART1/KS, Fuzzy ART, agrupamento difuso.

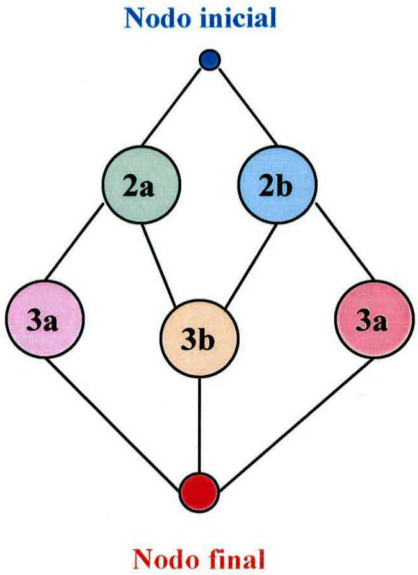


Figura 5.1 – Árvore de possibilidades para manufaturar uma peça requerendo dois tipos de máquinas (cada uma tendo duas unidades).

5.4 MINIMIZAÇÃO DA MOVIMENTAÇÃO INTERCELULAR CONSIDERANDO A SEQÜÊNCIA DE OPERAÇÕES

Nesta subseção, um modelo matemático é proposto para minimizar a movimentação intercelular no sistema de manufatura considerando a seqüência de operações das peças. Os movimentos das peças entre as máquinas de uma mesma célula (i.e. movimentos intracelulares) não são considerados no presente modelo.

As seguintes suposições são utilizadas pelos modelos apresentados neste capítulo:

1. Máquinas idênticas são atribuídas para a mesma célula. Sob este aspecto, nós consideramos somente o particionamento de "tipos de máquinas".
2. A demanda de produção das peças é conhecida e constante sob um determinado horizonte de planejamento. Cada tipo de peça possui um lote máximo transportável pelo sistema de transporte.
3. Cada tipo de peça pode ter múltiplas rotas de processamento (dependente do problema).
4. O número de células é conhecido.
5. Cada máquina pode executar somente uma operação por vez.

6. É admitido que um tipo de peça j possa requerer operações não-consecutivas na mesma máquina i.

7. O tempo da k-ésima operação realizada pela máquina i na peça j é conhecido e constante sob um determinado horizonte de planejamento.

8. A k-ésima operação requerida pelo tipo de peça j é realizada somente por uma máquina por vez. Não há separação do lote de peças do tipo j e a mesma operação sendo realizada por tipos de máquinas distintas.

10. Se duas operações consecutivas são necessárias na mesma máquina, então os tempos das operações são somados e considerados como uma só operação.

As seguintes notações são utilizadas pelos modelos:

I. Índices de Indexação:

$i = 1, 2,..., M$ (máquinas)

$j = 1, 2,..., N$ (peças)

$\{s,t\} = 1, 2,..., L$ (células)

$p = 1, 2,..., P_j$ (planos de processos)

II. Variáveis de Decisão:

$$X_{is} = \begin{cases} 1, \text{ se a máquina i está na célula s;} \\ 0, \text{ caso contrário.} \end{cases}$$

$$Y_{jp} = \begin{cases} 1, \text{ se a peça j utiliza o plano de processo p;} \\ 0, \text{ caso contrário.} \end{cases}$$

III. Parâmetros

M = Número total de tipos de máquinas existentes no sistema de manufatura.

N = Número total de tipos peças existentes no sistema de manufatura.

t_{ij} = Tempo de processamento unitário requerido para realizar a operação especificada na rota do tipo de peça j usando o tipo de máquina i. (tempo/unidade)

T_i = Tempo disponível do tipo de máquina i. (tempo/período)

N_j = Demanda de produção requerida para o tipo de peça j para um horizonte de planejamento pré-estabelecido. (unidade/período).

L_j = Dimensão do lote para o tipo de peça j. (unidade/lote)

Γ_j = Peso do tipo de peça j. Usualmente especifica o número de lotes movidos pelo sistema de transporte entre as células s e t. Neste caso,

$$\Gamma_j = \left[N_j / L_j \right] \text{ (lote/período)}$$

Pode também especificar somente a demanda do tipo de peça j, batelada ou o custo associado ao transporte.

Ψ_{stj} = Número de vezes que qualquer máquina pertencente à célula t é sucessora imediata de qualquer máquina pertencente à célula s na seqüência de operação do tipo de peça e vice-versa. (adimensional).

L = Número total de células.

LMC_{mim} = Número mínimo de máquinas por célula.

LMC_{max} = Número máximo de máquinas por célula.

Baseado no trabalho Harhalakis et al.[96], nós propomos o seguinte modelo matemático:

$$Z_1 = \sum_{s=1}^{L-1} \sum_{t=s+1}^{L} \sum_{j=1}^{N} \Psi_{stj} \cdot \Gamma_j . \tag{5.7}$$

Desta forma, o problema consiste em encontrar a partição $\widetilde{C}^* = \{C_1, C_2, ..., C_L\}$ que minimize Z_1, sujeito a

$$LMC_{min} \leq \sum_{i=1}^{M} X_{is} \leq LMC_{max}, \ \forall s \tag{5.8}$$

$$\sum_{s=1}^{L} X_{is} = 1, \ \forall i \tag{5.9}$$

$$X_{is} \in \{0,1\} \ (\forall i,s) \tag{5.10}$$

$$LMC_{min} < LMC_{max} \tag{5.11}$$

$$\sum_{p=1}^{P_j} Y_{jp} = 1 \ (\forall j) \qquad (5.12)$$

A restrição (5.8) assegura que todas as células contenham um número de máquinas dentro de um intervalo especificado pelo usuário. A restrição (5.9) assegura que cada máquina é atribuída para uma e somente uma célula (i.e., não é permitido que duas células compartilhem a mesma máquina). A restrição (5.10) garante que as variáveis de decisão serão valores binários e a restrição (5.11) assegura que os parâmetros LMC_{min} e LMC_{max} são diferentes. A restrição (5.12) assegura que somente um plano (rota) de processo é atribuído para um tipo de peça. Esta restrição vale somente quando o problema considera múltiplos planos de processos. A codificação das peças na representação composta dos indivíduos utilizada neste trabalho favorece sempre a satisfação da restrição (5.12): cada peça recebe um único valor na codificação, e, portanto, uma única rota é atribuída para cada peça.

5.4.1 ILUSTRAÇÃO DO MODELO Z_1

A fim de ilustrar o modelo Z_1, considere o agrupamento de 7 máquinas dentro das 3 células ilustrado na figura 5.2 (O_{jk} especifica em qual máquina o tipo de peça j realiza sua k-ésima operação). Neste exemplo, cada tipo de peça possui somente um plano de processo. A célula C_1 contém 3 máquinas (M_1, M_3 e M_5) e as células C_2 e C_3 contém 2 máquinas cada (M_2 e M_4, M_6 e M_7 respectivamente). Há 6 tipos de peças que são produzidas neste sistema de manufatura e cada tipo de peça possui uma seqüência bem definida de operações: $r_1 = [M_1\ M_2\ M_1]$, $r_2 = [M_3\ M_1]$, $r_3 = [M_7\ M_6]$, $r_4 = [M_4\ M_2]$, $r_5 = [M_5\ M_3\ M_4]$ e $r_6 = [M_7\ M_4]$. A demanda de produção destes tipos de peças é dada pelo vetor N_j = [100, 250, 150, 90, 160, 125].

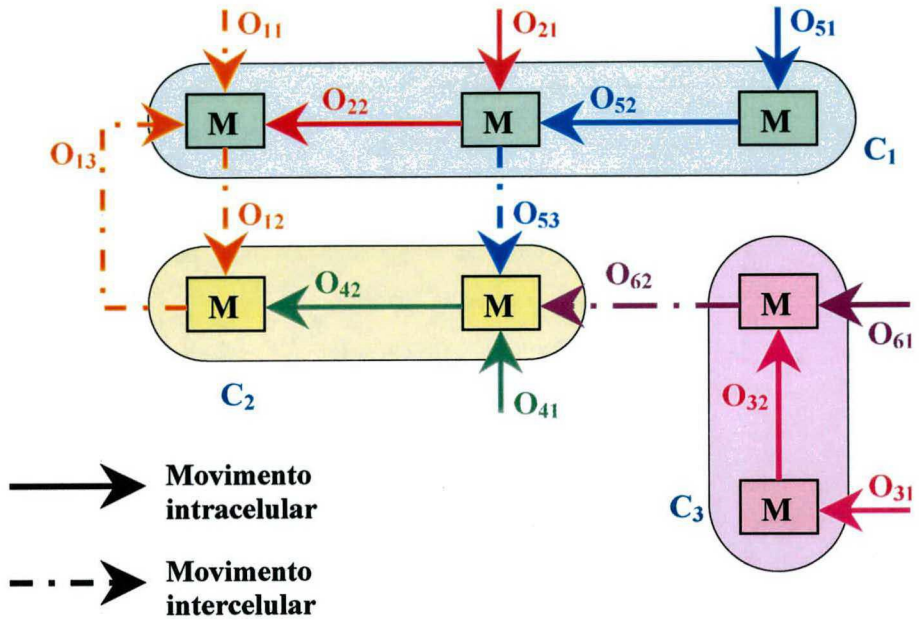
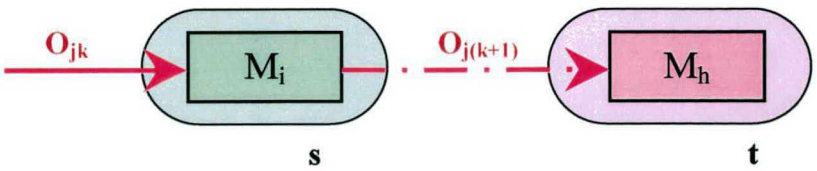


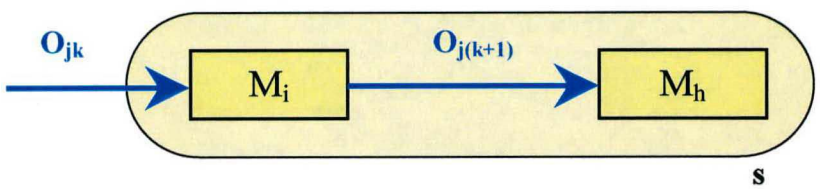
Figura 5.2 – Tráfego de peças entre as máquinas de um SMC.

Como pode ser visto na Figura 5.2, o tipo de peça 1 necessita ser transportado duas vezes entre as células C_1 e C_2, enquanto que os tipos de peças 4 e 5 necessitam de uma única movimentação intercelular (entre C_2 e C_3, C_1 e C_2 respectivamente).

Uma maneira para quantificar esta movimentação intercelular e incorporá-la em um modelo consiste em utilizar o parâmetro ψ_{stj} para contabilizar as viagens requeridas pelo tipo de peça j entre as células s e t. Esta idéia é ilustrada na Figura 5.3. Somente as operações consecutivas de uma peça j atribuídas às máquinas localizadas em células distintas (caso 1) podem incrementar o contador ψ_{stj}. Aquelas operações consecutivas realizadas dentro de uma mesma célula (caso 2) não são levadas em consideração, pois estamos interessados em avaliar somente a quantidade de movimentos intercelulares do sistema de manufatura.



(a) Caso 1



(b) Caso 2

Figura 5.3 – Atribuição de operações consecutivas para máquinas localizadas em células distintas (caso 1) e numa mesma célula (caso 2).

Considerando a demanda de produção requerida, a quantidade movida entre C_1 e C_2 do tipo de peça 1, por exemplo, é dado por $\psi_{121}*N_1 = 200$. Estendendo para todos os tipos de peças, a soma total de movimentos intercelulares no sistema de manufatura é igual a 415.

5.5 MINIMIZAÇÃO DA VARIAÇÃO DA CARGA DE TRABALHO NAS CÉLULAS

Este modelo, denominado aqui de Z_2, foi proposto por Venugopal e Narendran [8, 9] e serve para auxiliar no fluxo suave de materiais dentro de cada célula. Isto é, este modelo serve para minimizar a variação da carga total da célula entre as máquinas, sendo dado a seguir:

$$Z_2 = \sum_{i=1}^{M} \sum_{s=1}^{L} X_{is} \cdot \sum_{j=1}^{N} \left(w_{ij} - m_{sj}\right)^2 \qquad (5.13)$$

Portanto, nós desejamos minimizar Z_2 sujeito a

$$\sum_{s=1}^{L} X_{is} = 1 \ (\forall i) \tag{5.14}$$

$$\sum_{i=1}^{M} X_{is} \geq 2 \ (\forall s) \tag{5.15}$$

$$X_{is} = \{0,1\} \ (\forall i,s) \tag{5.16}$$

onde

$$w_{ij} = \left(t_{ij} \cdot N_j\right) / T_i \ , \tag{5.17}$$

$$m_{sj} = \left(\sum_{i=1}^{M} X_{is} \cdot w_{ij} \Big/ \sum_{i=1}^{M} X_{is} \right) . \tag{5.18}$$

Aqui, m_{sj} e w_{ij} representam a carga média da célula s induzida pela peça j e a carga de trabalho da máquina i induzida pela peça j, respectivamente.

5.6 UMA ABORDAGEM EVOLUTIVA BASEADA EM UM ALGORITMO GENÉTICO PARA RESOLVER OS MODELOS Z_1 E Z_2

Neste trabalho, nós propomos a resolução do PFCM baseada numa abordagem evolutiva, especificamente um AG, a fim de obter soluções quase-ótimas para os modelos propostos (Z_1 e Z_2). A seguir, nós detalhamos a abordagem evolutiva adotada.

5.6.1 REPRESENTAÇÃO E INICIALIZAÇÃO DA POPULAÇÃO P(t)

Nós devemos codificar as informações relevantes ao modelo matemático Z_1 e Z_2, i.e. as variáveis de decisão X_{is}. A forma de codificação utilizada é a mesma de Venugopal e Narendran [8] (ver subseção 2.8, Figura 2.21), pois esta representação é simples e eficiente para resolver o nosso problema.

Quando rotas de processamento alternativas são consideradas, a nossa representação é quase semelhante a de [70], com exceção que o valor do gene (parte das peças) expressa "que" rota foi atribuída para aquela peça e não "a quê" célula esta peça pertence (nós não faremos a formação simultânea de células de máquinas e famílias de peças).

A população é inicializada aleatoriamente, segundo uma determinada semente, mas um limite de diversidade é requerido para assegurar uma boa amostragem inicial do espaço de busca. Uma medida de diversidade é usada para este propósito, sendo computada da seguinte forma [8, 67]:

$$MD = \left(1/M\right) \sum_{i=1}^{M} \left[\frac{-\sum_{s=1}^{L} \left(n_{is}/N\right) \cdot \log\left(n_{is}/N\right)}{\log(L)} \right] \tag{5.19}$$

onde n_{is} representa o número de indivíduos na qual a máquina i é atribuída para a célula s na corrente população e N representa o tamanho da população utilizada pelo AG.

Nós consideramos que uma população inicial deve ter um valor de MD > 0.92. A medida de diversidade MD pode também ser utilizada durante o processo de evolução de P(t) para monitorar sua diversidade e, quando MD torna-se muito baixo, medidas para prevenir uma convergência prematura podem ser tomadas. Nós não utilizamos nenhuma heurística neste sentido em nossa abordagem. Repare que nossa população é criada de forma totalmente aleatória, o que pode ocasionar um grande número de indivíduos não-factíveis iniciais, i.e. que violam as restrições (5.8) ou (5.9). Nós não empregamos nenhum mecanismo de reparação nesta população inicial, apenas punimos aqueles indivíduos que são não-factíveis adicionando um termo punitivo muito alto ao valor da função objetivo destes indivíduos. Um típico mecanismo de reparação no nosso caso, se desejado, consistiria em testar se as restrições (5.8) ou (5.9) para cada indivíduo são satisfeitas. Se não, para cada célula irregular do indivíduo, uma realocação aleatória das máquinas excedentes (se o limite superior foi violado) ou faltantes (se o limite inferior foi violado) deveria ser levada a efeito. Esta realocação é realizada até que todas as células sejam válidas. Moon e Gen [91] utilizaram este procedimento.

5.6.2 OPERADORES DE VARIAÇÃO

Os operadores de variação utilizados são: (1) operador de recombinação de dois pontos e multipontos (i.e., mais de dois pontos de corte). Eles possuem um funcionamento igual aquele descrito nas seções 4.4. e 4.5. Em nossa abordagem, nós aplicamos os operadores em cada parte da representação (máquinas e peças, se esta última é considerada). O operador de recombinação multipontos é utilizado para os problemas que envolvem rotas alternativas de processamento (especificamente para o modelo Z_1).

O operador de mutação simplesmente muda o valor do gene dentro do intervalo definido pelo problema (células para máquinas e número de rotas para cada peça). A peça que possui somente uma rota de processamento é considerada no código genético, entretanto, ela não pode sofrer mutação.

5.6.3 PROCEDIMENTO DE BUSCA LOCAL

AGs são muito eficientes na exploração global do espaço de busca, entretanto, eles realizam uma busca local relativamente ineficiente, i.e., eles possuem certa dificuldade para encontrar a solução ótima local na região do espaço de busca para onde o AG convergiu (melhor solução encontrada) [78]. Desta forma, nós propomos um método heurístico de busca local estocástico (têmpera simulada, TS) para tentar aumentar a eficiência de nossa abordagem evolutiva, i.e. fazer um ajuste fino. Este algoritmo híbrido será denominado de AG/TS. A seguir, alguns detalhes de implementação relativos ao procedimento de busca local serão discutidos.

5.6.3.1 MECANISMO DE GERAÇÃO DA VIZINHANÇA $V(X_C)$

O movimento a partir de uma partição corrente X_c para uma outra partição vizinha X_v, tal que $X_v \in V(X_C)$, é obtido usando uma das seguintes transformações elementares:

(TE1) Uma máquina i é escolhida a partir de uma célula s e atribuída para outra célula t, sendo todo o processo aleatório. Esta troca de células não deve violar as restrições impostas aos modelos Z_1 e Z_2. A Figura 5.4 ilustra esta transformação elementar.

(TE2) Escolhe-se uma máquina i e h a partir de uma célula s e t, respectivamente, e efetua-se a troca simultânea destas máquinas. Novamente, as restrições impostas aos modelos não devem ser violadas. A figura 5.5 ilustra a segunda e última transformação elementar usada pela TS implementada neste trabalho (outras transformações, contudo, são possíveis).

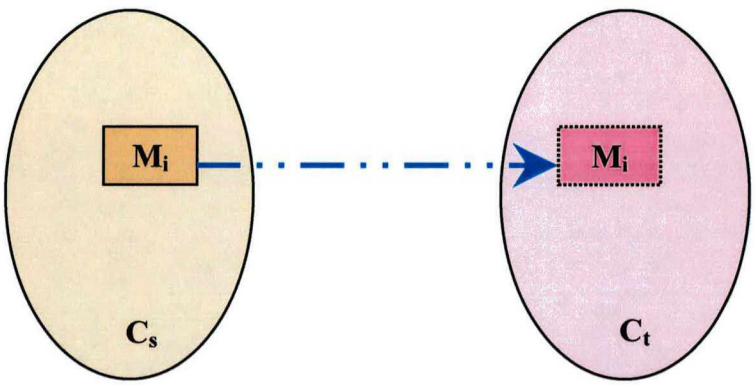


Figura 5.4 – Transformação elementar TE1.

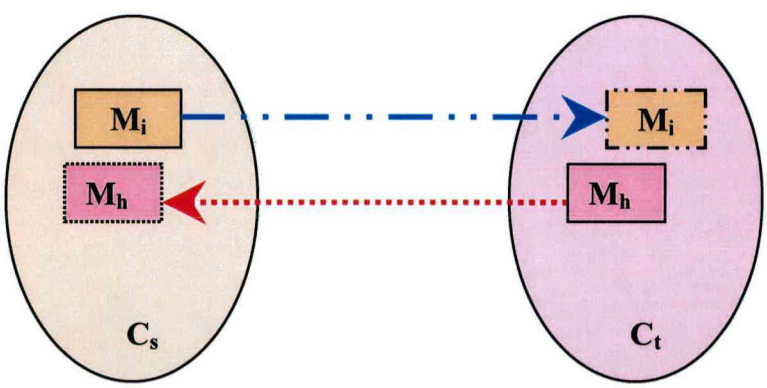


Figura 5.5 – Transformação elementar TE2.

Devemos enfatizar aqui que não fazemos qualquer busca especializada no código referente às peças (se estas são consideradas na representação).

5.6.3.2 PSEUDOCÓDIGO DA TÊMPERA SIMULADA

A TS implementada é baseada na heurística reportada por Vakharia e Chang [108], e é uma extensão do pseudocódigo apresentado na Figura 3.5 (ver subseção 3.4.2). A seguir, o pseudocódigo utilizado neste trabalho é apresentado (Figura 5.6).

```
t ← 0;
enquanto (t < T)
        t ← t + 1;
        s ← 0;
        enquanto (s < S_t)
                s ← s + 1;
                X_v ← gerar_vizinho(X_c,L,LIS);
                    Vobj_v ← avaliar_modelo(Z,X_p,L,S_p,N);
                se (Vobj_v - Vobj_c < 0) então
                    X_c ← X_v;
                    se (Vobj_v - Vobj_m < 0) então
                        X_m ← X_v;
                    fim_se
                senão se (Pr_a ≥ U(0,1))
                    X_c ← X_v;
                fim_se
        fim_enquanto
        Pr_a ← Pr_a - epsolon;
fim_enquanto
retornar {X_m, Vobj_m }
```

Figura 5.6 – Pseudocódigo da têmpera simulada (TS) baseado no trabalho de Vakharia e Chang [108].

5.6.3.3 PSEUDOCÓDIGO DO ALGORITMO GENÉTICO

O pseudocódigo do AG utilizado para resolver os dois modelos deste capítulo, Z_1 e Z_2, é mostrado na Figura 5.7. Outras funções objetivos podem ser empregadas no lugar de Z_1 e Z_2, mas as variáveis de decisão devem ser as mesmas, uma vez todo o AG é definido a partir da representação adotada.

O código da Figura 5.7 é inspirado na seguinte heurística: um indivíduo, especificamente o melhor da população, pode receber a chance de melhorar o seu valor de aptidão através de uma busca local. Uma heurística baseada na têmpera simulada é empregada (ver Figura 5.6). Entretanto, esta busca ocorre somente de tempos em tempos (aqui existe uma analogia com o intervalo de migração entre populações vizinhas, seção 4.8).

```
t ← 1;
criar P(t);
avaliar P(t);
enquanto {critério de parada == falso}
        selecionar Q(t) a partir de P(t);
        recombinar Q(t);
        mutar Q(t);
        avaliar Q(t);
        se {tempo de busca local == verdadeiro}
            executar TS(melhor indivíduo);
            reinserir melhor indivíduo em Q(t);
        fim_se
        reinserir Q(t) em P(t);
        t ← t + 1;
fim_enquanto
retornar {Q(t)}
```

Figura 5.7 – Pseudocódigo do AG/TS.

5.7 PROCEDIMENTO PARA FORMAÇÃO DE FAMÍLIAS DE PEÇAS

O algoritmo para identificar famílias de peças é descrito a seguir.

Passo 1: Ajustar j=1.

Passo 2: Determinar a família de peças para o qual a peça j é atribuída:

$F_s = \arg\{\max\{\sum_{i=1}^{M} a_{ij} \cdot X_{is}\}\}$ (a peça j é atribuída para a família de peças na qual a maioria de suas operações são feitas na correspondente célula de máquinas). Se a peça j requer a máquina i, então $a_{ij} = 1$, caso contrário, $a_{ij} = 0$.

Se o critério fornece um empate, pode-se utilizar o seguinte critério de desempate (não utilizado neste trabalho): a peça j é atribuída para a família de peças na qual ela possui o maior tempo de processamento na correspondente célula de máquinas.

Passo 3: Se {j = N} então pare, senão incremente j e retorne para o passo 2.

5.8 CONCLUSÃO

Neste capítulo, nós definimos formalmente o PFCM e o tamanho do seu espaço de busca. Dois modelos matemáticos para resolver o PFCM, que incluem importantes aspectos do chão-de-fábrica, foram apresentados e uma abordagem baseada em algoritmos genéticos foi proposta para resolver estes modelos. No capítulo seguinte, alguns problemas obtidos da literatura foram utilizados para validar a abordagem proposta neste capítulo.

CAPÍTULO 6

RESULTADOS EXPERIMENTAIS E DISCUSSÃO

6.1 INTRODUÇÃO

Neste capítulo, alguns problemas obtidos na literatura são utilizados para validar a abordagem proposta no capítulo anterior. Mas, antes de apresentarmos os resultados obtidos, é pertinente tecer alguns poucos comentários sobre a metodologia experimental adotada neste capítulo. Uma série de réplicas foram realizadas para cada problema, cada réplica tendo uma diferente semente para a geração dos números aleatórios utilizados pelas heurísticas AG e TS. Isto é necessário devido à natureza estocástica do AG e da TS. Nós somente podemos tirar observações conclusivas com respeito à performance de algoritmos estocásticos se olharmos os valores médios obtidos por eles (ou se um determinado valor, na média, ocorreu mais do que os outros). Duas sementes diferentes utilizadas em, por exemplo, um mesmo AG (i.e., ambos possuindo um conjunto de valores iguais para os seus parâmetros), tendem a resultar em comportamentos diferentes para ambos os AGs (mesmo que eles tenham obtido o mesmo valor da função objetivo no final).

Com respeito aos ajustes dos parâmetros-chaves utilizados pelas heurísticas AG e TS, não foi realizado nenhum estudo experimental mais acurado neste trabalho para determinar os seus melhores valores. Por exemplo, os principais parâmetros do AG, e.g. probabilidade de recombinação, probabilidade de mutação, tamanho da população, número de gerações e taxa de reinserção, foram determinados por tentativa e erro (a faixa de valores para estes parâmetros comumente reportada na literatura foi levada em consideração). Uma metodologia de teste que leve em consideração um estudo estatístico mais apurado e que permita uma melhor determinação destes parâmetros, para o caso do PFCM, é deixada como uma proposta para uma futura continuação deste trabalho. Como o enfoque principal deste trabalho é a utilização do AG em sua forma não híbrida, i.e. não executar o procedimento de busca local do algoritmo descrito na Figura 5.7 (ver subseção 5.6.3.3), nós utilizaremos o AG/TS em apenas alguns casos neste trabalho. Por fim, para implementar a abordagem proposta no capítulo 5, nós utilizamos algumas funções do *toolbox* de AGs do Matlab® (versão 4), desenvolvido por Chipperfield et al. [75], e implementamos os nossos

algoritmos na linguagem de programação *script* para a versão 5.2 do Matlab®. As funções utilizadas são: ***ranking.m*** (responsável por fazer o ranking de forma linear ou não-linear da população), ***reins.m*** (responsável por fazer a reinserção de uma parte da população da geração corrente na nova geração, de acordo com a taxa de reinserção) e ***sel_tour.m*** (responsável por fazer a seleção por torneio na população – modificada para admitir qualquer dimensão do torneio). Uma coisa importante a salientar aqui é que a reinserção foi realizada usando uma estratégia elitista (no problema 3 foi utilizada uma reinserção baseada em uma escolha aleatória). O resto da funcionalidade do AG teve que ser implementada levando em consideração as características da representação utilizada: simples quando somente máquinas são codificadas e composta quando máquinas e peças são codificadas juntas (ver subseção 5.6.1). A heurística TS também foi implementada no Matlab®. Portanto, todo o sistema desenvolvido neste trabalho foi codificado em arquivos denominados *m-files* (extensão .m) e são interpretados pelo Matlab®. Desta forma, a comparação entre o tempo de execução deste sistema com programas compilados não é correta, pois o código interpretado é mais lento do que o código compilado.

Os resultados obtidos neste trabalho serão apresentados a seguir. Alguns problemas serão descritos em maiores detalhes, outros apenas mencionados. O apêndice A contém os conjuntos de testes utilizados neste capítulo.

6.2 PROBLEMA 1: HARHALAKIS et al. [96]

O primeiro problema utilizado para validar o modelo Z_1 foi obtido em [96], sendo mostrado na tabela A.1. Este problema considera 20 tipos de peças e 20 tipos de máquinas.

A demanda de produção é unitária e somente operações não consecutivas são consideradas. Nós definimos o número mínimo de máquinas por célula igual a 2 (LMC_{mim}) e o número máximo de máquinas por célula igual a 7 (LMC_{max}). Estes valores foram adotados porque nós consideramos que uma célula deve ter ao menos 2 máquinas para justificar a sua formação e, quanto ao LMC_{max}, porque 7 foi o número obtido pelo algoritmo de Harhalakis em [96].

A tabela 6.1 apresenta os resultados obtidos para as 12 réplicas do problema 1, cada uma sendo executada com uma semente diferente e com os mesmos valores dos parâmetros-chaves do AG. Para a maioria das simulações, o valor da função objetivo é o mesmo daquele encontrado em [96], i.e. 14 movimentos intercelulares.

Tabela 6.1 – Resultados obtidos para as 12 réplicas do problema 1.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	150.50	14	26	20
2	150.71	14	19	15
3	150.55	16	21	14
4	150.56	14	25	18
5	151.37	15	25	26
6	150.44	14	23	14
7	150.99	16	22	39
8	150.88	14	21	18
9	150.57	14	23	20
10	151.49	14	22	18
11	151.27	15	21	29
12	150.67	14	22	25

[1] Relativo ao melhor valor encontrado.

Os parâmetros utilizados pelo o AG proposto são: NInd (número de indivíduos) = 100; ger (número de gerações) = 50; q (tamanho do torneio) = 8; $p_r = 0.7$ (probabilidade de recombinação – 2 pontos); p_m (probabilidade de mutação) = 0.1; GGAP (taxa de reinserção) = 0.95. O procedimento de busca local (TS) não foi utilizado.

As células de máquinas identificadas através de nossa abordagem são: C_1 = {2,3,5,11,14,16,17}; C_2 = {8,19,20}; C_3 = {1,9,10,12,18}; C_4 = {4,6,7,13,15}. As correspondentes famílias de peças são: F_1 = {2,4,6,7,11,15,19}; F_2 = {3,10,18}; F_3 = {1,9,12,14,17,20}; F_4 = {5,8,13,16}.

As famílias de peças foram obtidas usando o procedimento descrito na subseção 5.7. A Figura 6.1 ilustra a matriz de incidência com formação das células de manufatura obtidas através de nossa abordagem.

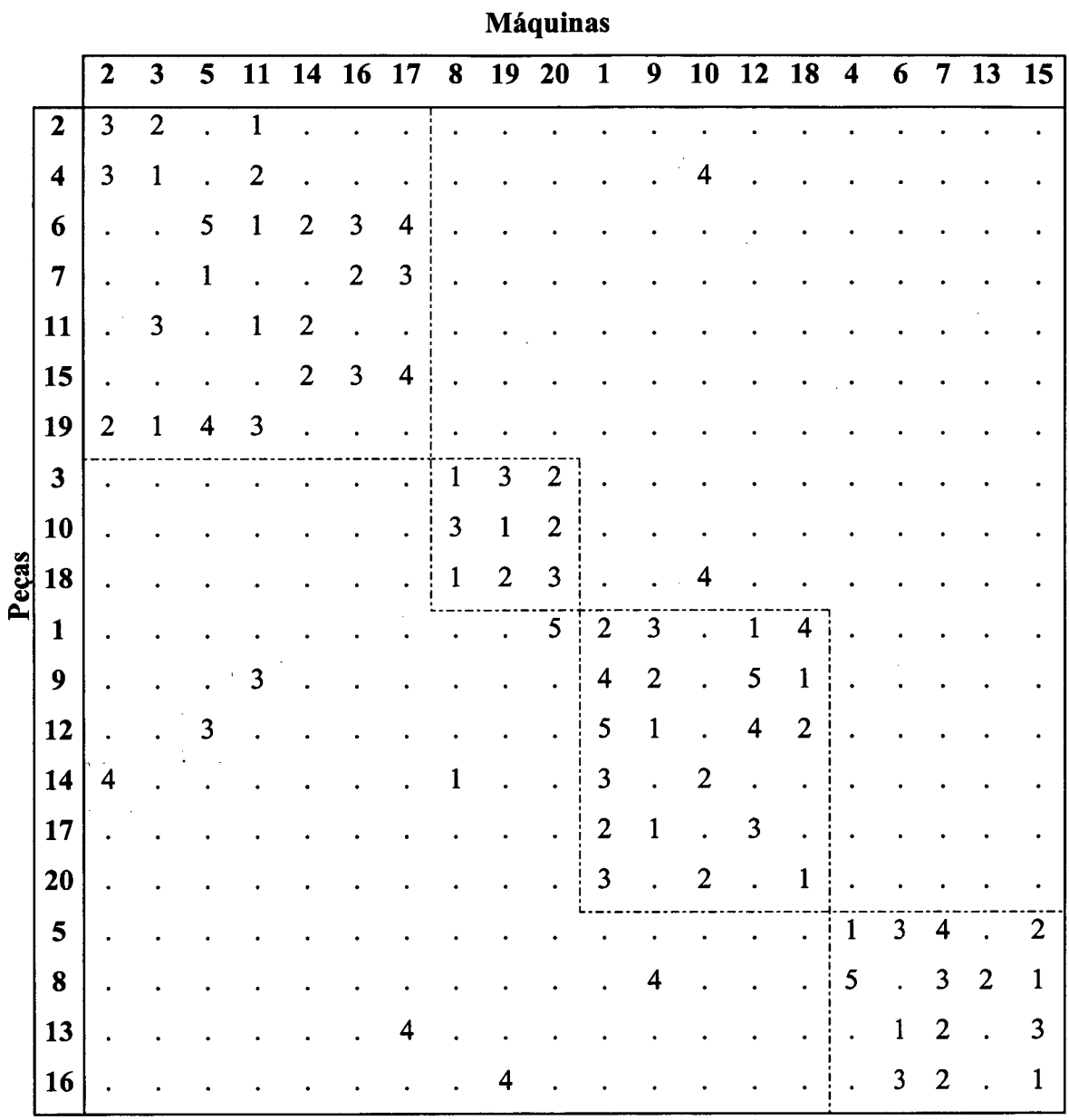


Figura 6.1 – Matriz de incidência máquina-peça do problema 1 contendo as células de manufatura formadas pelo AG proposto.

6.3 PROBLEMA 2: KAZEROONI et al. [64]

Este problema considera 13 tipos de peças, cada uma contendo 2 rotas de processamento, e 20 tipos de máquinas. A demanda de produção das peças não é unitária e algumas peças requerem operações não-consecutivas em certas máquinas. Neste problema, nós consideramos somente o número mínimo de máquinas por célula, sendo igual a 2.

A tabela 6.2 apresenta os resultados obtidos para as 12 réplicas do problema 2. Para a maioria das simulações, o valor da função objetivo é o mesmo daquele encontrado em [64], i.e. 65 unidades são movidas entre as células do sistema.

Tabela 6.2 – Resultados obtidos para as 12 réplicas do problema 2.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	66.29	65	2	28
2	66.84	65	5	30
3	66.85	65	1	31
4	66.13	65	2	33
5	66.24	195	4	34
6	66.26	65	2	28
7	66.74	155	1	19
8	66.30	65	6	24
9	66.30	155	2	22
10	66.57	65	1	23
11	66.79	65	6	24
12	66.41	65	2	22

[1] Relativo ao melhor valor encontrado.

Os parâmetros utilizados pelo o AG proposto são: NInd = 100; ger = 40; q = 2; p_r = 0.9 (recombinação multipontos); p_m = 0.1; GGAP = 0.95. As células de manufatura formadas são ilustradas na Figura 6.2.

Peças \ Máquinas	3	5	8	2	4	6	1	7
1	3	2	1	.	.	.	.	.
2	(1, 3)	2	.	.	.	.	.	.
4	2	3	1	.	.	.	.	.
7	.	2	1	.	.	.	.	.
9	1	3	2	.	.	.	.	.
5	.	.	.	1	.	2	.	.
6	.	.	.	1	2	.	.	.
8	.	.	.	.	1	2	.	.
10	.	.	.	3	1	(2, 4)	.	.
13	.	.	.	2	1	3	.	.
3	.	.	.	.	.	1	3	(2, 4)
11	.	.	.	.	.	.	2	(1, 3)
12	.	.	.	.	.	.	2	(1, 3)

Figura 6.2 – Matriz de incidência máquina-peça do problema 2 contendo as células de manufatura formadas pelo AG proposto.

6.4 PROBLEMA 3: KAZEROONI et al. [64]

Este é um problema relativamente difícil devido ao seu tamanho: possui 40 tipos de peças, sendo que as peças possuem diferentes números de rotas de processamento, e 30 tipos de máquinas. Há peças que necessitam de operações não-consecutivas em algumas máquinas. A tabela 6.3 apresenta os resultados obtidos para as 8 réplicas do problema 3.

O problema original conta com uma demanda de produção não-unitária, entretanto, nós tivemos uma certa dificuldade para resolve-lo considerando esta demanda. Quando nós consideramos uma demanda unitária, a maioria das simulações forneceu um valor igual a 1 (um movimento intercelular), sendo que as células de máquinas formadas foram iguais àquelas obtidas por Kazerooni et al. [64]. Nós definimos ainda que $LMC_{mim} = 4$ e $LMC_{max} = 7$.

Tabela 6.3 – Resultados obtidos para as 8 réplicas do problema 3.

Experimento	Tempo de execução (min.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	26.84	1	20	54
2	27.36	2	19	55
3	27.11	1	11	56
4	27.43	1	14	72
5	24.24	7	14	77
6	24.36	2	7	64
7	27.20	1	14	57
8	24.88	1	15	49

[1] Relativo ao melhor valor encontrado.

Os parâmetros utilizados pelo o AG proposto são: NInd = 200; ger = 100; q = 8; p_r = 0.6 (recombinação com multipontos); p_m = 0.03; GGAP = 0.98.

Outra interessante característica observada na resolução deste problema foi que os resultados não foram tão bons se uma reinserção usando uma estratégia elitista é empregada.

Para obter os resultados da tabela 6.3, foi adotada uma estratégia de reinserção aleatória. Isto pode provocar a perda do melhor indivíduo e, devido a isso, nós utilizamos uma memória externa que armazena o melhor indivíduo encontrado durante o processo evolutivo da população P(t).

6.5 PROBLEMA 4: ADAPTADO DE SURESH et al. [45]

Este problema foi adaptado da seguinte forma: no problema original todas as peças tinham somente uma rota de processamento e o valor da movimentação intercelular era igual a zero, i.e. a matriz máquina-peça possuía uma perfeita estrutura diagonal de blocos.

Nós incluímos uma rota alternativa para cada peça, sendo que algumas dessas rotas podem incluir operações que forçam certas peças a saírem de suas células para poderem ser totalmente manufaturadas, i.e. nós adicionamos um certo ruído ao problema original.

A tabela 6.4 apresenta os resultados obtidos para as 12 réplicas do problema 4.

Tabela 6.4 – Resultados obtidos para as 12 réplicas do problema 4.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	138.85	0	7	0
2	139.29	0	7	0
3	139.51	335	8	10
4	140.12	0	7	0
5	139.34	0	11	0
6	139.73	0	12	0
7	140.06	210	8	12
8	140.23	295	15	10
9	139.57	0	6	0
10	139.40	0	10	0
11	141.27	365	9	19
12	139.02	0	9	0

[1] Relativo ao melhor valor encontrado.

Através da tabela 6.4, percebe-se que para a maioria das réplicas, o valor da movimentação intercelular obtido é igual a zero, assim como acontece com o problema original [45]. Nós consideramos apenas que $LMC_{mim} = 2$ (nenhum limite superior é imposto ao AG).Os parâmetros utilizados pelo o AG são: NInd = 120; ger = 50; q = 8; $p_r = 0.9$ (recombinação com multipontos); $p_m = 0.1$; GGAP = 0.95. As células de manufatura formadas são ilustradas na Figura 6.3.

Máquinas

Peças	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	1	3	5	4	.	.	.	.	.	.	.	.	.	.
2	3	1	2	4	5	.	.	.	.	.	.	.	.	.	.
3	1	3	2	5	4	.	.	.	.	.	.	.	.	.	.
4	1	2	4	5	3	.	.	.	.	.	.	.	.	.	.
5	1	4	2	3	5	.	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	1	3	4	2	5	.	.	.	.	.
7	.	.	.	.	.	1	3	4	2	5	.	.	.	.	.
8	.	.	.	.	.	1	3	2	5	4	.	.	.	.	.
9	.	.	.	.	.	2	1	4	3	5	.	.	.	.	.
10	.	.	.	.	.	2	1	3	5	4	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.	1	3	2	5	4
12	.	.	.	.	.	.	.	.	.	.	1	2	5	3	4
13	.	.	.	.	.	.	.	.	.	.	2	3	1	4	5
14	.	.	.	.	.	.	.	.	.	.	1	2	4	5	3
15	.	.	.	.	.	.	.	.	.	.	1	3	2	5	4

Figura 6.3 – Matriz de incidência máquina-peça do problema 4 contendo as células de manufatura formadas pelo AG proposto.

6.6 PROBLEMA 5: SOUILAH et al. [79]

Para este problema, 19 tipos de peças são processadas em 12 máquinas. Souilah et al. [79] reportou um tráfego intercelular igual a 43 como o melhor resultado obtido pela sua abordagem (um AG com codificação binária). Os resultados obtidos através de nossa abordagem são mostrados na tabela 6.5.

Tabela 6.5 – Resultados obtidos para as 8 réplicas do problema 5.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	150.27	43	18	30
2	153.45	43	16	33
3	158.46	43	16	27
4	152.02	43	8	41
5	155.44	43	12	45
6	159.39	43	13	28
7	155.17	43	16	33
8	159.67	43	8	34

[1] Relativo ao melhor valor encontrado.

Os parâmetros utilizados pelo o AG são: NInd = 100; ger = 40; q = 2; p_r = 0.9 (recombinação com dois pontos); p_m = 0.1; GGAP = 0.95. Nós consideramos ainda LMC_{mim} = 2 e LMC_{max} = 4. As células de máquinas são: C_1 = {1, 2, 6}; C_2 = {4, 7, 8, 9}; C_3 = {3, 5}; C_4 = {10, 11, 12}. As famílias de peças formadas são: F_1 = {5, 11, 18}; F_2 = {1, 2, 3, 4, 6, 7, 8, 9, 10}; F_3 = {12, 13, 14, 15, 16, 17, 19}. Este é um inconveniente do método para formação de famílias de peças adotado: o número de famílias pode ser menor, como observado neste problema, do que o número de células. Desta forma, mais de uma célula de máquina é responsável por processar uma família de peças.

6.7 PROBLEMA 6: VENUGOPAL E NARENDRAN [8]

Para testar o modelo Z_2, nós utilizamos o problema reportado em [8]. Nós apenas trocamos a função objetivo Z_1 por Z_2, o que demonstra a flexibilidade de nossa abordagem.

A mesma configuração de células de manufatura reportada em [8] foi obtida para todas as réplicas deste experimento. Um fato interessante deste problema foi a ausência quase total de indivíduos não-factíveis na população final do nosso algoritmo.

Os parâmetros utilizados pelo o AG são: NInd = 100; ger = 40; q = 2; p_r = 0.9 (recombinação com dois pontos); p_m = 0.1; GGAP = 0.95.

Nós consideramos LMC_{mim} = 2 e não colocamos nenhum limite no número máximo de máquinas dentro de uma célula. Os resultados obtidos são mostrados na tabela 6.5 e a solução deste problema é mostrada na Figura 6.4.

Tabela 6.5 – Resultados obtidos para as 12 réplicas do problema 5.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	99.74	7.316	13	1
2	101.17	7.316	21	0
3	108.31	7.316	13	0
4	101.50	7.316	16	0
5	100.95	7.316	18	0
6	101.39	7.316	12	0
7	100.41	7.316	17	2
8	101.23	7.316	12	0
9	101.39	7.316	13	0
10	101.28	7.316	13	0
11	100.51	7.316	18	0
12	100.51	7.316	7	0

[1] Relativo ao melhor valor encontrado.

	Peças																													
	1	3	8	9	6	7	10	11	19	17	12	14	16	18	5	20	22	26	27	2	4	13	21	23	24	25	15	28	29	30
1	.	.3	.2	.2	.6	.6	.5	.7	.6	.4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	.4	.5	.4	.3	.7	.3	.6	.8	.2	.9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	.6	.7	.9	.6	.2	.4	.2	.2	.5	.3	.	.	.4	.	.	.	.	.	.	.	.3	.	.	.	.	.	.	.	.	.
7	.8	.9	.5	.7	.3	.5	.3	.5	.9	.6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.6	.2	.2	.3	.3	.9	.4	.5	.8	.6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.4	.5	.7	.8	.3	.9	.6	.8	.2	.8	.	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.	.	.	.	1	.7	.2	.3	.9	.4	.5	.6	.8	1.1	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.	.	.7	.6	.2	.3	.4	.4	.5	.6	.2	.	.4	.	.	.	.	.5	.	.	.
8	.	.	.	.	.	.	.	.	.	.	.3	.8	.3	.9	1.2	.2	.3	.4	.5	.2	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.	.6	.9	.5	.6	.5	.7	.8	.9	1	4	.	.	.	.	.	.	.	.	.	.
11	.3	.3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.2	.3	.5	.9	.2	.5	.4	.6	.7	.8
12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.6	.7	.9	.9	.3	.5	.8	.5	.6	.7
13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.7	.5	.8	.5	.3	.4	.6	.5	.7	.8
14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.2	.6	1	.5	.4	.6	.8	.8	.2	.8
15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.5	.7	.	.3	.	.7	.9	.9	.3	.4

Figura 6.4 – Células de manufatura formadas através da minimização do modelo Z_2.

6.8 PROBLEMA 7: KAMAL E BURKE [46]

Os parâmetros utilizados para resolver este problema são os mesmo do problema 5, sendo que novamente nós consideramos $LMC_{mim} = 2$. Este problema foi originalmente resolvido por Kamal e Burke [46] utilizando uma rede neural FART (ver subseção 2.6.2). Nós calculamos a variação de carga (modelo Z_2) obtida pela solução dos autores, a qual é igual a 14.514. O valor obtido através de nossa abordagem é menor do que o valor obtido em [46], para todas as réplicas.

As células de máquinas obtidas pelo AG são: $C_1 = \{1, 2, 3, 10, 11, 12, 13\}$; $C_2 = \{9, 14\}$; $C_3 = \{6, 8\}$; $C_4 = \{4, 5, 7\}$. Para efeito de comparação, as células obtidas em [46] foram: $C_1 = \{1, 12, 13\}$; $C_2 = \{2, 3, 10, 11\}$; $C_3 = \{4, 5, 7\}$; $C_4 = \{6, 8, 9, 14\}$.

A diferença entre a solução encontrada por Kamal e Burke e a nossa é a seguinte: a heurística FART agrupa por similaridade, i.e., ela minimiza o movimento intercelular mas não a variação da carga dentro das células.

Nós empregamos uma heurística própria para isso e, devido a isto, o valor encontrado pela nossa solução é menor (o valor da movimentação intercelular da nossa solução é

maior do que em [46]). Os objetivos de minimizar o movimento intercelular e a variação da carga dentro das células são conflitantes.

Tabela 6.6 – Resultados obtidos para as 12 réplicas do problema 6.

Experimento	Tempo de execução (seg.)	Melhor valor encontrado	Indivíduos ótimos[1]	Indivíduos não-factíveis
1	102.44	13.1401	9	15
2	101.50	13.1401	9	19
3	102.00	13.1401	12	27
4	101.66	13.1401	3	14
5	101.17	13.4977	6	13
6	101.94	13.4977	1	13
7	102.98	13.1401	13	12
8	104.12	13.1401	10	14
9	101.67	13.7088	8	14
10	100.95	13.1401	7	26
11	101.23	13.1401	10	20
12	102.60	13.1401	15	19

[1] Relativo ao melhor valor encontrado.

6.9 DISCUSSÃO DOS RESULTADOS

Os problemas apresentados nas seções anteriores apresentaram diferentes graus de dificuldade. Problemas envolvendo rotas alternativas de processamento e a seqüência de operações não-consecutivas em uma mesma máquina foram consideradas. Nós podemos observar que o AG proposto conseguiu, ou igualar os resultados obtidos (problemas 1, 2, 3, 4, 5 e 6), ou se saiu melhor (problema 7), quando comparados com os resultados obtidos pelos autores originais. Nós observamos que um parâmetro crítico do AG para a abordagem proposta foi o tamanho do torneio (q). Em alguns problemas, nós tivemos que utilizar q = 8. O problema com relação a isto é que, quanto maior o tamanho do torneio, mais rápido ocorre a perda de diversidade da população, podendo resultar em uma permanência em mínimos locais (o AG fica estagnado em uma solução de baixa qualidade e não consegue mais diminuir a função objetivo, tanto Z_1 quanto Z_2) . Tradicionalmente, o tamanho do

torneio reportado na literatura é 2, mas isto não é uma regra. Entretanto, para muitos problemas nós conseguimos utilizar este valor. É interessante mencionar que nós utilizamos uma pressão seletiva igual a 2 e um ranking linear no nosso AG. Testes preliminares com um ranking não-linear (neste caso a pressão seletiva pode ser maior do que 2) não demonstraram nenhuma melhora significativa.

O problema 3 não foi satisfatoriamente resolvido quando a demanda de produção das peças é utilizada como dados de entrada. Os resultados obtidos ficaram cerca de 10% acima (na média) do valor obtido originalmente pelos autores. Entretanto, quando esta demanda é omitida, o nosso AG conseguiu resolver o problema muito bem. Devemos enfatizar que nenhum processo de busca local foi realizado neste problema. Mesmo não utilizando a demanda de produção, o problema continua difícil de se resolver devido a quantidade de escolhas de rotas a serem feitas para as peças.

Com respeito ao termo de penalidade, nós escolhemos um valor de 1000 quando a demanda de produção é unitária e 10000 quando ela não é (para os problemas 1, 2 ,3 e 4).

É interessante notar que para alguns problemas, o número de indivíduos não-factíveis foi alto. Isto ocorreu principalmente para os problemas onde os limites inferiores e superiores foram definidos, LMC_{mim} e LMC_{max} respectivamente. Desta forma, operações de recombinações multipontos entre dois indivíduos, mesmo sendo os dois factíveis, possuem uma considerável probabilidade de resultarem pelo menos um indivíduo não-factível. A recombinação multiponto, contudo, permite manter uma boa diversidade da população durante o processo de otimização.

Os problemas 6 e 7, eles não representaram nenhuma dificuldade para serem resolvidos, mesmo tendo um número médio de máquinas a serem agrupadas. Sintonizar os valores do nosso AG para resolver eles foi muito fácil, principalmente para o problema 6.

Com relação aos problemas 1, 2, 4 e 7, nós pegamos algumas réplicas cujos valores estavam acima daqueles reportados na literatura e empregamos o algoritmo híbrido AG/TS para resolvê-los. Nós definimos o parâmetro TBL – acrônimo de tempo de busca local – e fizemos com que, no máximo, ocorresse duas vezes o procedimento de busca local durante o processo evolutivo. Para fazer isto, nós implementamos a seguinte regra no teste "se" do algoritmo 5.7 (pág. 112): quando o resto da divisão do número de gerações corrente por TBL for igual a zero, então ocorre a busca local por TS.

Nós utilizamos quase os mesmos parâmetros reportados em [108], exceto que o valor da época foi bem maior. Conseqüentemente, o custo computacional também foi maior,

praticamente o triplo do AG. Como resultados, nós conseguimos melhorar razoavelmente o valor da função objetivo, principalmente para os problemas 1 e 7. Entretanto, a eficiência do AG/TS para resolver os problemas compostos (2 e 4) foi comprometida devido ao fato de somente estarmos fazendo uma busca local em relação ao código das máquinas. Os valores dos parâmetros utilizados pela TS são: $Pr_0 = 0.5$ (probabilidade de aceitação inicial), $T = 20$ (número total de iterações) e $S_T = 200$ (número de buscas em cada iteração).

CAPÍTULO 7

CONCLUSÃO E PERPECTIVAS FUTURAS

Com respeito aos problemas testados, nós obtivemos bons resultados, comparáveis àqueles obtidos originalmente pelos autores. Nós utilizamos dois modelos que incorporaram importantes aspectos do chão-de-fábrica, principalmente a seqüência de operações e a inclusão de rotas alternativas de processamento. A nossa implementação do AG mostrou ser bastante eficiente e flexível, pois nós utilizamos os dois modelos seqüencialmente sem fazer nenhuma modificação. Isto somente foi possível devido ao fato que os dois modelos utilizaram as mesmas variáveis de decisões, as quais são codificadas como cadeias de números inteiros. Outros modelos que incorporam características diferentes daquelas assumidas neste trabalho, mas que utilizem esta mesma representação de inteiros, podem ser utilizados com nenhum ou pouco esforço de implementação.

As contribuições deste trabalho podem ser enumeradas da seguinte forma:

1. Uma revisão geral e atual do estado-da-arte dos métodos utilizados na formação de células de manufatura. Diversos exemplos e conceitos foram apresentados, com uma grande ênfase na exposição de diversos algoritmos de propósito geral.
2. O desenvolvimento de um modelo simples (Z_1) que leva em consideração rotas alternativas para as peças, demanda de produção e seqüência de operações (inclusive operações de peças não consecutivas em uma mesma máquina).
3. O desenvolvimento de uma abordagem baseada em algoritmos genéticos e sua hibridização, inclusive demonstrando sua flexibilidade ao poder utilizar um outro modelo (Z_2), sem que nenhum esforço adicional fosse preciso.

Como sugestões para trabalhos futuros, nós listamos alguns tópicos que tendem a contribuir com a continuação deste trabalho:

1. Resolver os modelos utilizados neste trabalho simultaneamente, utilizando uma das abordagens listadas no capítulo 3. A otimização multi-objetivo e sua aplicação ao PFCM ainda são um tópico muito pouco estudado e poucos trabalhos foram reportados na década de 90. Uma valiosa contribuição para a área seria a otimização simultânea dos dois modelos apresentados neste trabalho usando um AEMO (algoritmo evolutivo multi-objetivo).

2. Elaboração de modelos mais sofisticados. O apêndice B apresenta um exemplo utilizando um modelo Z_2 que estende aquele modelo desenvolvido em [8, 9] e que foi discutido no capítulo 5. Ele trabalha com rotas alternativas e com operações não-consecutivas em uma mesma máquina. Outro ponto relevante neste tópico seria o desenvolvimento de modelos mais complexos que incluam não somente características de manufatura, mas também características de projetos das peças. A consideração e o tratamento de máquinas duplicadas é altamente desejável numa possível evolução deste trabalho.
3. Um estudo sobre a formação simultânea de células de máquinas e de família de peças.
4. Realizar um trabalho experimental, juntamente com um estudo estatístico, que permita determinar uma faixa de valores ótima para os parâmetros utilizados pelo AGs (com respeito aos modelos apresentados neste trabalho). Uma boa sintonia dos parâmetros dos AGs é de vital importância para se resolver eficientemente um POC, sem que as soluções freqüentemente sejam de péssima qualidade.
5. A paralelização da implementação proposta é um estudo altamente recomendável. Uma área em aberto seria a utilização dos modelos distribuídos comentados no capítulo 3. Mas este já é um tópico muito mais complexo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Groover, M. *Automation, Production Systems, and Computer Integrated Manufacturing*. Prentice Hall, Englewood, NJ, 1987.

[2] Singh, N. Design of cellular manufacturing systems – an invited review. *European Journal of Operational Research*, v. 69, n. 3, p. 284-291, Sep. 1993.

[3] Heragu, S. S. Group technology and cellular manufacturing. *IEEE Transactions on Systems, Man and Cybernetics*, v. 24, n. 2, p. 203-215, Feb. 1994.

[4] Hyer, N. L., Wemmerlöv, U. Group technology and productivity. Harvard Business Review, v. 62, n. 4, p. 140-149, 1984.

[5] ———— Group technology in the United-States manufacturing industry – a survey of current practices. *International Journal of Production Research*, v. 27, n. 8, p. 1287-1304, Aug. 1989.

[6] Wemmerlöv, U.; Hyer, N. L. Cellular manufacturing in the US Industry: A survey of Users. *International Journal of Production Research*, v. 27, n. 9, p. 1511-1530, Sep. 1989.

[7] Wemmerlöv, U.; John, D. Cellular Manufacturing at 46 users plant: implementations experiences and performance improvements. *International Journal of Production Research*, v. 35, n. 1, p. 29-49, 1997.

[8] Venugopal, V.; Narendran, T. T. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers & Industrial Engineering*, v. 22, n. 4, p. 469-480, Oct. 1992.

[9] ———— Cell-formation in manufacturing systems through simulated annealing – an experimental evaluation. *European Journal of Operational Research*, v. 63, n. 3, p. 409-422, Dec. 1992.

[10] Ballakur, A.; Steudel, H. A within-cell utilization based heuristic for designing cellular manufacturing systems. International Journal of Production Research, v. 25, n. 5, p. 639-655, 1987.

[11] Aguiar, M. S. *Análise Formal da Complexidade de Algoritmos Genéticos*. Porto Alegre, 1998. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade do Rio Grande do Sul.

[12] Joines, J. A.; King, R. E.; Culbreth, C. T. Comprehensive review of production-oriented cell formation techniques. *International Journal of Flexible Automation and Integrated Manufacturing*, v. 3, n. 3-4, p. 161-200, 1996.

[13] Aurélio, B. *Novo Aurélio: O Dicionário da Língua Portuguesa.* Editora Fronteira, 1984.

[14] Pearl, J. *Heuristics : Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, Massachusetts, 1984.

[15] Back, T.; Hammel; U.; Schwefel, H.-P. Evolutionary computation: comments on the history and current state. *IEEE Transactions On Evolutionary Computation*, v. 1, n. 1, p. 3-17, Apr. 1997.

[16] Wemmerlöv, U.; Hyer, N. L. Research issues in cellular manufacturing. *International Journal of Production Research*, v. 25, n. 3, p. 413-431, 1987.

[17] Chu, C.-H.; Tsai, M. A comparison of three array-based clustering techniques for manufacturing cell formation. *International Journal of Production Research*, v. 28, n. 8, p. 1417-1433, 1990.

[18] Cheng, C.H. Algorithms for grouping machine groups in group technology. *International Journal of Management Science*, v. 20, 493-501, 1992.

[19] King, J. R. Machine-component group formation in group technology: review and extension. *International Journal of Production Research*, v. 18, n. 2, p. 213-232; 1980.

[20] Chang, T.-C.; Wysk, R.; Wang, H.-.P. *Computer-Aided Manufacturing.* Prentice Hall, 1997.

[21] King, J. R.; Nakornchai, V. Machine-component group formation in group technology: review and extension. *International Journal of Production Research*, v. 20, n. 2, p. 117-133; 1982.

[22] Chan, H. M.; Milner, D. A. Direct clustering algorithm for group formation. *Journal of Manufacturing Systems*, v. 1, n. 1, 65-74 (1982).

[23] McCormick, W. T.; Schweiter P.J., White, T.W. Problem decomposition and data reorganization by a cluster technique. *Operations Research*, v.20, n.5, 993-1009, 1972.

[24] Lenstra, J. K., Clustering a data array and the traveling salesman problem. Operations Research, v.22, 413-414, 1974.

[25] Gupta, T. Clustering algorithms for the design of a cellular – an analysis of their performance. *Computers & Industrial Engineering*, v. 20, n.4, 343-353, 1991.

[26] Carpenter, G. A.; Grossberg, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, n. 37, 54-115, 1987.

[27] Carpenter, G. A.; Grossberg, S. The ART of adaptive pattern recognition by a self-organizing neural network. Computer, n. 31, 77-88, Mar. 1988.

[28] Fausett, L. V. *Fundamentals of Neural Networks*, Prentice Hall, 1994.

[29] Freeman, J. A.; Skapura, D. M. *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley Publishing Co., 1991.

[30] Carpenter, G. A.; Grossberg, S. ART 2: stable self-organization of pattern recognition codes for analog input patterns. *Applied Optics*, 26, 4919-4930, 1987.

[31] Carpenter, G. A.; Grossberg, S.; Rosen, D. B. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, v. 4, 759-771, 1991b.

[32] Carpenter, G. A.; Grossberg, S.; Rosen, D. B. Artmap. ART3 – Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, v. 3, p. 129-152, 1990.

[33] Carpenter, G. A.; Grossberg, S.; Reynolds, J. H. ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, v. 4, p. 565-588, 1991.

[34] Carpenter, G. A.; Grossberg, S.; Markuzon, N. et al. Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. Technical Report CAS/CNS-91-016. Boston, MA: Boston University.

[35] Suresh, N. C.; Kaparthi, S. Performance of fuzzy ART neural network for group technology cell-formation. *International Journal of Production Research*, v. 32, n. 7, p. 1693-1713, Jul. 1994.

[36] Dagli, C.; Huggahalli, R. Machine-part family formation with the adaptive resonance theory paradigm. . *International Journal of Production Research*, v. 33, n. 4, p. 893-913, 1995.

[37] Kusiak, A.; Chung, Y. GT/ART: using neural networks to form machine cells. Manufacturing Review, v. 4, n. 4, 293-301, 1991.

[38] Kaparthi, S.; Suresh, N. C. Machine component cell-formation in group technology – neural network approach. *International Journal of Production Research*, v. 30, n. 6, p. 1353-1367, Jun. 1992.

[39] Kaparthi, S.; Suresh, N. C.; Cerveny, R. P. An improved neural-network leader algorithm for part-machine grouping in group technology. *European Journal of Operational Research*, v. 69, n. 3, p. 342-356, Sep. 1993.

[40] Chen, D. S.; Chen, H. C., Park, J. M. An improved ART neural net for machine cell formation. *Journal of Materials Processing Technology*, v. 61, n. 1-2, 1-6, 1996.

[41] Enke, D; Ratanapan, K.; Dagli, C. Machine-part family formation utilizing an ART1 neural network implemented on a parallel neuro-computer. *Computers & Industrial Engineering*, v. 34, n. 1, p. 189-205, Jan. 1998.

[42] Venugopal, V.; Narendran, T.T. Machine-cell formation through neural network models. *International Journal of Production Research*, v. 32, p. 2105-2116, 1994.

[43] Chen, S.-J.; Cheng, C.-S. A neural network-based cell formation in group technology. *International Journal of Production Research*, v. 33, n. 2, p. 293-318, 1995

[44] Liao, W. T.; Chen, L. An evaluating of ART1 neural models for GT part family and machine cell forming. *Journal of Manufacturing Systems*, v. 12, n. 4, p. 282-290, 1993.

[45] Suresh, N. C.; Slomp, J.; Kaparthi, S. Sequence-dependent clustering of parts and machines: a Fuzzy ART neural network approach. *International Journal of Production Research*, v. 37, n. 12, p. 2793-2816, Aug. 1999.

[46] Kamal, S.; Burke, L. I. FACT: a new neural network-based clustering algorithm for group technology. *International Journal of Production Research*, v. 34, n. 4, p 919-946, Apr. 1996.

[47] Kohonen, T. Analysis of a simple self-organizing process. *Biological Cybernetics*, v. 44, 135-140, 1982.

[48] Kohonen, T. Self-organized formation of topologically correct feature maps *Biological Cybernetics*, v. 43, 56-69, 1982.

[49] Kohonen , T. Self-organizing maps. *IEEE Special*, 12, 1990.

[50] Favata, F.; Walker, R. A study of the application of Kohonen-type neural networks to the travelling salesman problem. *Biological Cybernetics*, v. 64, 463-468, 1991.

[51] Kaski, S. *Data exploration using self-organizing maps*. Finland, 1997. Tese (Doutorado em Tecnologia da Informação). Helsinki University of Technology.

[52] Jang. I. H.; Rhee, J. T. Generalized machine cell formation considering material flow and plant layout using modified self-organizing feature maps. *Computers & Industrial Engineering*, v. 33, n. 3-4, p. 457-460, Dec. 1997.

[53] Kulkarni, U. R.; Kiang, M. Y. Dynamic grouping of parts in flexible manufacturing systems – a self-organizing neural networks approach *European Journal of Operational Research*, v. 84, n. 1, p. 192-212, Jul. 1995.

[54] Nour, M.A.; Madey, G. R. Heuristic and optimization approaches to extending the Kohonen self organizing algorithm *European Journal of Operational Research*, v. 93, n. 2, p. 428-448, Sep. 1996.

[55] Bezdek J. C.; Harthaway, R. J. Optimization of fuzzy clustering criteria using genetic algorithms. In: IEEE CONFERENCE ON EVOLUTIONARY COMPUTATION (First : June 1994 : Orlando, Florida). v. 2, p. 589-595.

[56] Hall, L. O.; Bezdek, J. C.; Boggavarpu et. al. Genetic fuzzy clustering. In: INTERNATIONAL JOINT CONFERENCE OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY (First : December 1994 : San Antonio, Texas), p. 411-417.

[57] Bezdek, J. C. *Pattern Recognition With Fuzzy Objective Function Algorithms*, Plenum Pub Corp, june 1981.

[58] Xu, H.; Wang, H. P. Part family formation for GT aplications based on fuzzy mathematics. *International Journal of Production Research*, v. 27, n. 9, p 1637-1651, 1989.

[59] Hon, K. K. B.; Chi, H. A new approach of group technology in part family optimization. *Annals of Tech CIRP*, 43, 425-428, 1994.

[60] Kazerooni, M.; Luong, L. H. S.; Abhary, K. Machine chain similarity, a new approach for cell formation. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED PRODUCTION ENGINEERING (11th : London, England). *Proceedings*. p. 97-105.

[61] ———— A genetic Algorithm based approach to cell formation. In: INTERNATIONAL CONFERENCE ON MANUFACTURING ENGINEERING (6th : 29 Nov. – 1 Dec. 1995 : Melbourne, Australia). *Proceedings*. p. 615-622.

[62] Kazerooni, M.; Luong, L. H. S.; Abhary, K. Cell formation using genetic algorithms. International Journal of Flexible Automation and Integrated Manufacturing, v. 3, n. 3-4, p. 283-299, 1995.

[63] Kazerooni, M.; Luong, L. H. S.; Abhary, K. et al. An integrated method for cell layout problem using genetic algorithms. In: INTERNATIONAL CONFERENCE ON CAD/CAM ROBOTICS AND FACTORIES OF THE FUTURE (12th : 14 – 16 Aug. 1996 : London, England). *Proceedings*. p. 752-762.

[64] Kazerooni, M.; Luong, L. H. S.; Abhary, K. A genetic algorithm based cell design considering alternative routing. *Computer Integrated Manufacturing Systems*, v. 10, n. 2, p. 93-107, May 1997.

[65] Pereira, V. L. D. V. *Aplicação do algoritmo genético na formação de célula de manufatura*. Florianópolis, 1994. Tese (Doutorado em Engenharia Mecânica) – Departamento de Engenharia Mecânica, Universidade Federal de Santa Catarina.

[66] Paris, J. L.; Pierreval, H. Manufacturing cell formation using distribuited evolutionary algorithms. In: INTERNATIONAL CONFERENCE ON CAD/CAM ROBOTICS AND FACTORIES OF THE FUTURE (12th : 14 – 16 Aug. 1996 : London, England). *Proceedings*. p. 369-374.

[67] Gupta, Y.; Gupta, M.; Kumar, A. et al. A genetic algorithm-based approach to cell composition and layout design problems. *International Journal of Production Research*, v. 34, n. 2, p. 447-482, Feb. 1996.

[68] Logendran, R. Impact of operations and layout of cells in cellular manufacturing. *International Journal of Production Research*, v. 29, n. 2, p 375-390, Feb. 1991.

[69] Hwang, H.; Sun, J. U. A genetic-algorithm-based heuristic for the GT cell formation problem. *Computers & Industrial Engineering*, v. 30, n. 4, p. 941-955, Sep. 1996.

[70] Joines, J. A.; Culbreth, C. T.; King, R. E. Manufacturing cell design: an integer programming model employing genetic algorithms. *IIE Transactions*, v. 28, n. 1, p. 69-85, Jan. 1996.

[71] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. AI Series, Springer-Verlag, NewYork, 1994.

[72] Morad, N.; Zalzala, A.M.S. The formation of manufacturing cells using genetic algorithms. In: ANNUAL CONFERENCE OF THE IRISH MANUFACTURING

COMMITTEE (12th : 6-8 Sept. 1995 : University College, Cork, Ireland). *Proceedings*.

[73] Chipperfield, A.; Fleming, P. J.; Pohlheim, H. et al. Genetic algorithm toolbox for use with Matlab. Technical Report n. 512. Department of Automatic Control and Systems Engineering, University of Sheffield, UK, 1994.

[74] Al-Sultan, K. S.; Fedjki, C. A. A genetic algorithm for the part family formation problem. *Production Planning & Control*, v. 8, n. 8, p. 788-796, 1997.

[75] Kusiak, A. The generalized group technology concept. *International Journal of Production Research*, v. 25, n. 4, p. 561-569, Apr. 1987.

[76] Joines, J. A.; Kay., M. G.; King, R. E. A hybrid-genetic algorithm for manufacturing cell design. IIE Transactions in review.

[77] Ng, S. Worst-case analysis of an algorithm for cellular manufacturing. *European Journal of Operational Research*, v. 69, n. 3, p. 384-398, 1993.

[78] Houck, C. R.; Joines, J. A.; Kay, M. G. et al. Empirical investigation of the benefits of partial Lamarckianism. *Evolutionary Computation*, v. 5, n. 1, p. 31-60, 1996.

[79] Souilah, A.; Chergui, M.-A.; Boulif, M. et al. Manufacturing cell formation in group technology with cohabitation constraints: a genetic lgorithms approach. In: IFAC/IFIP CONFERENCE ON MANAGEMENT AND CONTROL OF PRODUCTION AND LOGISTICS – MCPL'97 (31 Aug.- 3 Sep. 1997 : Campinas, SP, Brazil). *Proceedings*. p. 516-522.

[80] Gravel, M.; Nsakanda A. L.; Price, W. Efficient solutions to the cell-formation problem with multiple routings via a double-loop genetic algorithm. *European Journal of Operational Research*, v. 109, n. 2, p. 286-298, Sep. 1998.

[81] Cheng, C. H.; Gupta, Y. P.; Lee WH et al. A TSP-based heuristic for forming machine groups and part families. *International Journal of Production Research*, v. 36, n. 5, p. 1325-1337, 1998.

[82] Whitley, D. The GENITOR algorithm and selective pressure: why rank-based allocation of reproductive trials is best. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. (3th : 1989). Morgan Kaufmann, 1989. p. 116-121.

[83] Pierreval, H.; Plaquin, M.-F. An evolutionary approach of multicriteria manufacturing cell formation. International Transactions Operational Research, v. 5, n. 1, p. 13-25, 1998.

[84] Horn, J.; Nafpliotis, N. Multiobjective optimization using the niched Pareto genetic algorithm, Technical Report IlliGAl 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[85] Horn, J.; Nafpliotis, N.; Goldberg, D. E. A niched Pareto genetic algorithm for multiobjective optimization. In: IEEE CONFERENCE ON EVOLUTIONARY COMPUTATION, IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE (First : June 1994 : Piscataway, New Jersey). IEEE Service Center, 1994. v. 1, p. 82-87.

[86] Su, C. T.; Hsu, C. M. Multi-objective machine-part cell formation through parallel simulated annealing. *International Journal of Production Research*, v. 36, n. 8, p. 2185-2207, Aug. 1998.

[87] Dimopoulos, C.; Mort, N. Genetic programming for cellular manufacturing. In: WESIC'99 CONFERENCE (Sep. 1999 : Newport, Gwent). *Proceedings*.

[88] Mak, K. L.; Wong, Y. S.; Wang, X. X. An adaptive genetic algorithm for manufacturing cell formation. *International Journal of Advanced Manufacturing Technology*, v. 16, n. 7, p. 491-497, 2000.

[89] Plaquin, M. F.; Pierreval, H. Cell formation using evolutionary algorithms with certain constraints. *International Journal of Production Economics*, v. 64, n. 1-3, p. 267-278, Mar. 2000.

[90] De Lit, P.; Falkenauer, E.; Delchambre, A. Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Mathematics and Computers in Simulation*, v. 51, n. 3-4, p. 257-271, Jan. 2000.

[91] Moon, C.; Gen, M. A genetic algorithm-based approach for design of independent manufacturing cells. *International Journal of Production Economics*, v. 60-61, p. 421-426, Apr. 1999.

[92] Moon, C.; Kim, J. Genetic algorithm for maximizing the parts flow within cells in manufacturing cell design. *Computers & Industrial Engineering*, v. 36, n. 2, p. 379-389, Apr. 1999.

[93] Balakrishnan, J.; Jog, P. D. Manufacturing cell-formation using similarity coefficients and a parallel genetic TSP algorithm – formulation and comparison. *Mathematical and Computer Modelling*, v. 21, n. 12, p. 61-73, Jun. 1995.

[94] Dimopoulos, C.; Zalzala, A. M. S. Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation*, v. 4, n. 2, p. 93-113, 2000.

[95] Joines, J. A.; King, R. E.; Culbreth, C. T. Cell formation using genetic algorithms and applications in North Carolina furniture industries. In: Group Technology and Cellular Manufacturing, Boston : Kluwer Academic Publishers, 1998. p. 53-169.

[96] Harhalakis, G.; Nagi, R.; Proth, J. M. An Efficient heuristic in manufacturing cell-formation for group technology applications. *International Journal of Production Research*, v. 28, n. 1, p. 185-198, Jan. 1990.

[97] Harhalakis, G.; Proth, J. M.; Xie, X. L. Manufacturing cell design using simulated annealing – na industrial application. *Journal of Intelligent Manufacturing*, v. 1, n. 3, p. 185-191, Sep. 1990.

[98] Logendran, R. A binary integer programming approach for simultaneous machine-part grouping in cellular manufacturing systems. *Computers & Industrial Engineering*, v. 24, n. 3, p. 329-336, Jul. 1993.

[99] Nair, G. J.; Narendran, T. T. CASE: a clustering algorithm for cell formation with sequence data. *International Journal of Production Research*, v. 36, n. 1, p. 157-179, Jan. 1998.

[100] Heragu, S. S.; Gupta, Y. P. A heuristic for designing cellular manufacturing facilities. *International Journal of Production Research*, v. 32, n. 1, p. 125-140, Jan. 1994.

[101] Akturk, M. S.; Balkose, H. O. Part-machine grouping using a multi-objective cluster analysis. *International Journal of Production Research*, v. 34, n. 8, p. 2299-2315, Aug. 1996.

[102] Selim, H. M.; Askin, R. G.; Vakharia, A. J. Cell formation in group technology: review, evaluation and directions for future research. *Computers & Industrial Engineering*, v. 34, n. 1, p. 3-20, Jan. 1998.

[103] Abdelmola, A. I.; Taboun, S. M. Productivity model for the cell formation problem: a simulated annealing algorithm. *Computers & Industrial Engineering*, v. 37, n. 1-2, p. 327-330, Oct. 1999.

[104] Sofianopoulou, S. Application of simulated annealing to a linear model for the formulation of machine cells in group technology. *International Journal of Production Research*, v. 35, n. 2, p. 501-511, Feb. 1997.

[105] Boctor, F. F. The minimum-cost, machine-part cell formation problem. *International Journal of Production Research*, v. 34, n. 4, p. 1045-1063, 1996.

[106] Caux, C.; Bruniaux, R.; Pierreval H. Cell formation with alternative process plans and machine capacity constraints: a new combined approach. *International Journal of Production Economics*, v. 64, n. 1-3, p. 279-284, Mar. 2000.

[107] Chen, W.-H.; Srivastava, B. Simulated annealing procedures for forming machine cells in group technology. *European Journal of Operational Research*, v. 75, n. 1, p. 100-111, May 1994.

[108] Vakharia, A. J.; Chang, Y. L., Cell formation in group technology: a combinatorial search approach. *International Journal of Production Research*, v. 35, n. 7, p. 2025-2043, Jul. 1997.

[109] Zhou, M.; Askin, R. G. Formation of general GT cells: an operation-based approach. *Computers & Industrial Engineering*, v. 34, n. 1, p. 147-157, Jan. 1998.

[110] Zolfaghari, S.; Liang, M. Machine cell/part family formation considering processing times and machine capacities: a simulated annealing approach. *Computers & Industrial Engineering*, v. 34, n. 4, p. 813-823, Sep. 1998.

[111] Dahel, N.-E.; Smith, S. B. Designing flexibility into cellular manufacturing systems. *International Journal of Production Research*, v. 31, n. 4, p. 933-945, Jul. 1993.

[112] Hwang, C.-L.; Masud, A. S. M. *Multiple Objective Decision Making - Methods and Applications*. Lecture Notes in Economics and Mathematical Systems. New York, Springer-Verlag, 1979.

[113] Fonseca, C. M.; Fleming, P. J. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. (5^{th} : 1993 : San Mateo, California). *Proceedings*. Morgan Kauffman Publishers, 1993. p. 416-423.

[114] Van Veldhuizen, D. A.; Lamont, G. B. Multiobjective Evolutionary Algorithms: analyzing the state-of-the-art. *Evolutionary Computation*, v. 8, n. 2, p. 125-147, 2000.

[115] Srinivas, N.; Deb, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, n. 3, p. 221-248, Fall 1994.

[116] Fonseca, C. M.; Fleming, P. J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: a unified formulation. *IEEE*

Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, v. 28, n. 1, p. 26-37, 1998.

[117] Fonseca, C. M. *Multiobjective genetic algorithms with applications to control engineering problems*. Sheffield, UK, Sep. 1995. Tese (Doutorado em Engenharia de Sistemas). Department of Automatic Control and Systems Engineering, University of Sheffield.

[118] Zitzler, E. *Evolutionary algorithms for multiobjective optimization: methods and applications*. Zürich, Dec. 1999. Tese (Doutorado em Engenharia da Computação). Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich.

[119] Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 4, p. 257-271, Nov. 1999.

[120] Zitzler, E.; Thiele, L. An evolutionary algorithm for multiobjective optimization: the strength Pareto approach. Technical Report n. 43. Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich. May 1998.

[121] Looi, C. Neural network methods in combinatorial optimization. *Computers and Operations Research*, v. 19, n.3-4, p. 191-208, 1992.

[122] Cichocki, A.; Unbehauen, R. *Neural networks for Signal Processing and Optimization*. Wiley & Sons, 1994.

[123] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley Publishing Co., 1989.

[124] Fogel, D. B. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.

[125] Glover, F. Tabu search - part I. *ORSA Journal of Computing*, v.1, n. 3, p. 190-206, 1989.

[126] Kirpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671-680, 1983.

[127] Durbin, R.; Willshaw, D. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, v. 326, n. 16, p. 689, Apr. 1987.

[128] Blickle, T. *Theory of evolutionary algorithms and application to system synthesis*. Zürich, 1996. Tese (Doutorado em Engenharia da Computação) - Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich.

[129] Pirlot, M. General local search methods. *European Journal of Operational Research*, v. 92, p. 493-511, 1996.

[130] Metropolis, N.; Rosenbluth, A., Rosenbluth, M. et al. Equation of state calculation by fast computing machines. *Journal of Chemistry Physics*, v. 21, p. 1087-1092, 1953.

[131] Cerny, V. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, v. 45, p. 41-51, 1985.

[132] Colorni, A.; Dorigo, M.; Maffioli, F. et al. Heuristics from nature for hard combinatorial optimization problems. *International Transactions on Operational Research*, v. 3, n. 1., p. 1-21. 1996.

[133] Wang, T.-Y.; Lin, H.-C.; Wu, K.-B. An improved simulated annealing for facility layout problems in cellular manufacturing systems. *Computers & Industrial Engineering*, v. 34, n. 2, p. 309-319, 1998.

[134] Souilah, A. Simulated annealing for manufacturing systems layout design. *European Journal of Operational Research*, v. 82, p. 592-614, 1995.

[135] Koza, J. R. *Genetic Programming : On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. MIT Press, Dec. 1992

[136] Koza, J. R. The genetic programming paradigm: genetically breeding populations of computer programs to solve problems. In Soucek, Branko and the IRIS Group (editors). *Dynamic, Genetic, and Chaotic Programming*. New York: John Wiley. Pages 203-321, 1992.

[137] Wolpert, D. H.; Macready, W. G. No free lunch theorems for optimization, *IEEE Transactions On Evolutionary Computation*, v. 1, n. 1, p. 67-82, Apr. 1997.

[138] Culberson, J. On the futility of blind search: an algorithmic view of "No Free Lunch". *Evolutionary Computation Journal*, v. 6, n. 2, p. 109-128, 1998.

[139] Bittencourt, G. *Inteligência Artificial: Ferramentas e Teorias*. Florianópolis: Editora da Universidade Federal de Santa Catarina, 1998.

[140] De Jong, K. Genetic Algorithms: A 25 Year Perspective. In: IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE (June 26-July 2, 1994). *Proceedings*.

[141] Coello, C. A. C. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, v. 1, n. 3, p. 269-308, Aug. 1999.

[142] Srinivas, M; Patnaik, L. M. Genetic algorithms: a survey. Computer, v. 27, n. 6, p. 17-26, june 1994.

[143] Pohlheim, H. GEATbx: Genetic and Evolutionary Algorithm Toolbox for us with MATLAB. Disponível em: http://geatbx.com/index.html. Último acesso feito em abril de 2002.

[144] Beasley, D.; Bull, D.; Martin, R. R. An overview of genetic algorithms: part I, fundamentals. *University Computing*, v. 15, n. 2, p. 58-69, 1993.

[145] Bäck, T.; Hoffmeister, F. Extended Selection Mechanisms in Genetic Algorithms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. (4^{th} : 1991 : San Mateo, California, USA). *Proceedings*. Morgan Kaufmann Publishers, 1991. p. 92-99.

[146] Blickle, T. and Thiele, L. A comparison of selection schemes used in genetic algorithms. Technical Report n. 11, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich. Dec. 1995.

[147] Mühlenbein, H.; Dirk Schlierkamp-Voosen, D. The science of breeding and its application to the breeder genetic algorithm BGA. *Evolutionary Computation*, 4, p. 335-360, 1994.

[148] Bäck, T. Generalized convergence models for tournament and (μ,1)-selection. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, (6^{th} : 1995 : San Francisco, CA). *Proceedings*. Morgan Kaufmann, CA, 1995. pp. 2-8.

[149] Srinivas, M.; Patnaik, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, v. 24, n. 4, p. 656-666, 1994.

[150] Grefenstette, J. J. Optimization of control parameters for genetic algorithms. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-16, n. 1, p. 122-128, 1986.

[151] Michalewicz, Z., Heuristic methods for evolutionary computation techniques. *Journal of Heuristics*, v. 1, n. 2, p.177-206, 1995.

[152] Michalewicz, Z.; Dasgupta, D.; Le Riche, R.G. et al. Evolutionary algorithms for constrained engineering problems, *Computers & Industrial Engineering Journal*, v. 30, n. 2, p.851-870, Sep. 1996.

[153] Michalewicz, Z.; Schoenauer, M.. Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation*, v. 4, n. 1, p.1-32, 1996.

[154] Coello, C. A. C. A survey of constraint handling techniques used with evolutionary algorithms. Technical Report Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada, 1999.

[155] Richardson, J.T.; Palmer, M.R.; G. Liepins et al. Some guidelines for genetic algorithms with penalty functions. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS (3rd : 1989). *Proceedings*. Morgan Kaufmann, CA, 1989. pp. 191-197.

[156] Coit, D. W.; Smith, A. E.; Tate, D. M. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, v. 8, n. 2, p. 173-182, Spring 1996.

[157] Lienig, J. Parallel genetic algorithm for performance-driven VLSI routing. *IEEE Transactions On Evolutionary Computation*, v. 1, n. 1, p. 29-39, Apr. 1997.

[158] Cantú-Paz, E. *A Survey of Parallel Genetic Algorithms*. Illinois Genetic Algorithms Laboratory, Revised version of IlliGAL Report n. 95007, 28 pp, 1997.

[159] Tanomaru, J. Motivação, fundamentos e aplicações de algoritmos genéticos. In: II CONGRESSO BRASILEIRO DE REDES NEURAIS (II. : 29 Out. – 1 Nov. 1995 : Curitiba, Paraná). pp. 373-403.

[160] Husbands, P. Distributed coevolutionary genetic Algorithms for multi-criteria and multi-constraint optimisation. In: Evolutionary Computing, Lecture Notes in Computer Science Springer Verlag, v. 865, 1994. p. 150-165.

[161] Mühlenbein, H.; Dirk Schlierkamp-Voosen, D. Analysis of selection, mutation, and recombination operators in genetic algorithms. In: *Evolution as a Computational Process, Lecture Notes in Computer Science*. Berlim: Springer-Verlag, 1995. p. 188-214.

[162] Graham, R.; Patashnik, O.; Knuth, D. E. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Pub Co, March 1994.

APÊNDICE A – CONJUNTOS DE DADOS UTILIZADOS NO CAPÍTULO 6

Tabela A.1 – Dados do 1º problema [96]:
20 peças e 20 máquinas.

Peças	Demanda	Máquinas
1	1	12, 1, 9, 18, 20
2	1	11, 3, 2
3	1	8, 20, 19
4	1	3, 11, 2, 10
5	1	4, 15, 6, 7
6	1	11, 14, 16, 17, 5
7	1	5, 16, 17
8	1	15, 13, 7, 9, 4
9	1	18, 9, 11, 1, 12
10	1	19, 20, 8
11	1	11, 14, 3
12	1	9, 18, 5, 12, 1
13	1	6, 7, 15, 17
14	1	8, 10, 1, 2
15	1	13, 14, 16, 17
16	1	15, 7, 6, 19
17	1	9, 1, 12
18	1	8, 19, 20, 10
19	1	3, 2, 11, 5
20	1	18, 10, 1, 12

Tabela A.2 – Dados do 2º problema [64]:

13 peças com 2 rotas e 8 máquinas.

Peças	Demanda	Rotas	Máquinas
1	100	1	8, 5, 3
		2	6, 3
2	80	1	3, 5, 3
		2	2, 5, 1
3	65	1	6, 7, 1, 7
		2	2, 5, 2, 1
4	120	1	8, 3, 5
		2	6, 3, 6, 3
5	120	1	2, 1, 7
		2	2, 6
6	170	1	2, 4
		2	1, 4, 5
7	85	1	8, 7, 8, 7
		2	8, 5
8	90	1	4, 1
		2	4, 6
9	110	1	3, 8, 5
		2	2, 1, 5, 2
10	100	1	1, 3, 2, 1
		2	4, 6, 2, 6
11	125	1	7, 3, 4, 3
		2	7, 1, 7
12	120	1	5, 2, 1
		2	1, 7, 1
13	110	1	4, 2, 6
		2	7, 1,7

Tabela A.3 – Dados do 3º problema [64]:

40 peças com número de rotas variadas e 30 máquinas.

Peças	Demanda	Rotas	Máquinas
1	155	1	8, 9, 15, 16, 19, 20
		2	8, 30, 28, 2
2	160	1	1, 22, 24, 30
		2	10, 12, 13, 17
3	135	1	7, 12, 16, 13
		2	12, 13 ,17
		3	9, 13, 17, 30
4	150	1	6, 11, 1, 4, 7
		2	1, 2, 22, 24
		3	6, 11, 28, 30
5	210	1	3, 4, 5, 23
		2	6, 11, 5
		3	7, 26, 30
6	230	1	26, 28, 30
		2	12, 14, 20, 22
7	85	1	3, 4, 5, 23, 27, 29
8	90	1	7, 30, 8, 2
		2	7, 26, 30
		3	9, 26, 5, 11, 2
9	95	1	8, 10
		2	6, 7, 26
		3	1, 27, 4
10	86	1	12, 13, 17
		2	13, 19, 17
		3	8, 9, 12
11	55	1	18, 19, 22
		2	9, 15, 16
		3	12, 13, 17
12	120	1	12, 13, 17

Peças	Demanda	Rotas	Máquinas
		2	8, 12, 9, 30, 7
13	142	1	8, 9, 16, 19, 20
14	140	1	11, 18
		2	7, 12, 16, 13
15	100	1	2, 22, 24
		2	9, 26, 5, 2
16	65	1	11, 2, 9
		2	9, 15, 16, 19, 20
17	85	1	13, 19, 17
		2	10, 14, 11, 13
		3	1, 2, 22, 24
18	125	1	8, 9, 15, 16, 19, 20
		2	8, 9, 15, 24, 2
19	102	1	1, 22, 24
20	105	1	10, 13, 17
21	75	1	1, 22, 16, 13
		2	1, 22, 24
		3	6, 14, 18
22	100	1	1, 2, 21, 22, 24
		2	8, 9, 15, 24, 2
		3	9, 5, 11, 2
23	140	1	8, 9, 24, 2
		2	1, 22, 24
		3	1, 22, 24, 20
		4	7, 30, 8, 5
24	62	1	11, 14, 18
25	85	1	7, 26, 28, 30
		2	7, 30, 12, 17
26	185	1	8, 30, 12, 17
		2	14, 18, 15, 18
		3	12, 15, 6, 12

Peças	Demanda	Rotas	Máquinas
		4	10, 12, 17
27	55	1	6, 14, 18, 12, 13
		2	6, 11, 14, 18
28	130	1	22, 24, 5, 22
		2	1, 2, 22, 24
29	125	1	1, 2, 22, 24
30	135	1	26, 8, 5, 4
		2	7, 28, 30
31	65	1	12, 5, 8, 9
		2	13, 15, 6, 3
		3	10, 12, 13, 17
32	90	1	3, 4, 23, 27
		2	3, 24, 23, 6
33	100	1	26, 28, 30
		2	5, 8, 9, 21
34	90	1	12, 13, 17, 5, 2
		2	3, 4, 5, 23, 25, 27
35	120	1	9, 15, 19, 20
36	130	1	3, 4, 25, 27, 29
		2	6, 11, 28, 30
		3	8, 12, 9, 30
37	145	1	13, 19, 17
		2	3, 4, 5, 23, 25, 29
		3	10, 14, 11, 13
38	250	1	7, 26, 30
		2	10, 11, 10, 13
39	60	1	6, 15, 9, 6
		2	6, 11, 18
40	90	1	7, 26, 28

Tabela A.4 – Dados do 4º problema, adaptado de [45]:

15 peças com 2 rotas e 15 máquinas.

Peças	Demanda	Rotas	Máquinas
1	60	1	2, 1, 3, 5, 4
		2	2, 6, 2
2	75	1	2, 3, 1, 4, 5
		2	4, 13, 14
3	135	1	1, 3, 2, 5, 4
		2	5, 4, 9
4	100	1	1, 2, 5, 3, 4
		2	3, 2, 3
5	120	1	1, 3, 4, 2, 5
		2	3, 4, 2, 7
6	135	1	6, 9, 7, 8, 10
		2	7, 9, 7, 8
7	95	1	6, 9, 7, 8, 10
		2	6, 4, 5, 8
8	105	1	6, 8, 7, 10, 9
		2	3, 2, 9
9	80	1	7, 6, 8, 10, 9
		2	8, 7, 10, 11
10	75	1	7, 6, 8, 10, 9
		2	8, 7, 10, 11
11	100	1	11, 13, 12, 15, 14
		2	9, 12, 11, 12
12	120	1	11, 12, 14, 15, 13
		2	13, 12, 14, 12
13	140	1	13, 11, 12, 14, 15
		2	8, 3, 11, 13

Tabela A.4 – Continuação.

Peças	Demanda	Rotas	Máquinas
14	105	1	11, 12, 15, 13, 14
		2	14, 2, 12
15	80	1	11, 13, 12, 15, 14
		2	9, 8, 11, 9

Tabela A.5 – Dados do 5º problema [79]: 19 peças e 8 máquinas.

Peças	Batelada	Máquinas
1	2	1, 4, 8, 9
2	3	1, 4, 7, 4, 8, 7
3	1	1, 2, 4, 7, 8, 9
4	3	1, 4, 7, 9
5	2	1, 6, 10, 7, 9
6	1	6, 10, 7, 8, 9
7	2	6, 4, 8, 9
8	1	3, 5, 2, 6, 4, 8, 9
9	1	3, 5, 6, 4, 8, 9
10	2	4, 7, 4, 8
11	3	6
12	1	11, 7, 12
13	1	11, 12
14	3	11, 7, 10
15	1	1, 7, 11, 10, 11, 12
16	2	1, 7, 11, 10, 11, 12
17	1	11, 7, 12
18	3	6, 7, 10
19	2	12

Máquinas \ Peças	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	.	.	.3	.	.	.6	.6	.2	.2	.5	.7	.	.	.	.	.	.4	.	.6	.	.	.	.	.	.	.	.	.	.	.
2	.4	.	.5	.	.	.7	.3	.4	.3	.6	.8	.	.	.	.	.	.9	.	.2	.	.	.	.	.	.	.	.	.	.	.
3	.6	.	.7	.3	.	.2	.4	.9	.6	.2	.2	.	.	.	.	.4	.3	.	.5	.	.	.	.	.	.	.	.	.	.	.
4	.	.2	.	.	.3	.	.	.	.	.	.	.	.4	.7	.5	.6	.	.2	.	.4	.	.4	.	.	.	.5	.6	.	.	.
5	.	.2	.	.	.3	.	.	.	.	.	.	.4	.	.5	.	.7	.	.8	.	.9	.	.6	.	.	.	.8	.2	.	.	.
6	.	.8	.	.	.9	.	.	.	.	.	.	1	.	.7	.	.2	.	.3	.	.4	.	.5	.	.	.	.6	.8	.	.	.
7	.8	.	.9	.	.	.3	.5	.5	.7	.3	.5	.	.	.	.	.	.6	.	.9	.	.	.	.	.	.	.	.	.	.	.
8	.	1.1	.	.	1.2	.	.	.	.	.	.	.3	.	.8	.	.3	.	.9	.	.2	.	.3	.	.	.	.4	.5	.	.	.
9	.	.4	.	.	.5	.	.	.	.	.	.	.6	.	.9	.	.5	.	.6	.	.7	.	.8	.	.	.	.9	1	.	.	.
10	.6	.	.2	.	.	.3	.9	.2	.3	.4	.5		.	.	.	.	.6	.	.8	.	.	.	.	.	.	.	.	.	.	.
11	.3	.	.3	.2	.	.	.	.	.	.	.	.	.3	.	.4	.	.	.	.	.	.5	.	.9	.2	.5	.	.	.6	.7	.8
12	.	.	.	.6	.	.	.	.	.	.	.	.	.7	.	.8	.	.	.	.	.	.9	.	.9	.3	.5	.	.	.5	.6	.7
13	.	.	.	.7	.	.	.	.	.	.	.	.	.5	.	.6	.	.	.	.	.	.8	.	.5	.3	.4	.	.	.5	.7	.8
14	.	.	.	.2	.	.	.	.	.	.	.	.	.6	.	.8	.	.	.	.	.	1	.	.5	.4	.6	.	.	.8	.2	.8
15	.	.	.	.5	.	.	.	.	.	.	.	.	.7	.	.9	.	.	.	.	.	.	.	.3	.	.7	.	.	.9	.3	.4

Figura A.1 – Dados do 6º problema [8]: Matriz de incidência máquina-peça (30 peças e 15 máquinas).

Máquinas \ Peças	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	.	.	.	1	.	.69	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	.	.	.7	.	.	.7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	.	.	.22	.5	.	.2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.33	.	.	.
4	1	1	.	.	.3	.	.	.	.	.	.	.	.	.	.	.	.26	.	1	.74	.	.	.65	.
5	.36	.7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	1	.
6	.	.	.	.	.	.	.	.	.83	.73	1	1	.	1	.8	.66	.	.	.	.	.	.88	.	.
7	.72	.	.27	.	.	.	1	.	.	.	.	.	.	.	.	.	.5	.	.	.5	.	.	.	.
8	.	.	.	.	1	.	.	.	.57	1	.	.85	.97	.3	.45	1	.	.	.	.	.	1	.	.
9	.	.	.	.	.8	.	.	.	.28	.	.	.	.	.	.72	.	.	.	.	.	.	.	.	.
10	.	.	.46	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11	.	.	1	.82	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.76
12	.	.	.	.	.	.	.24	.93	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.7	.03	1	.	.	.	.	.	.	.	.	.	.68	.	.	.	.	.46	.
14	.	.	.	.	.	.	.	.	1	.	.24	.	1	.	1	.	.	.	.	.	.	.	.3	.

Figura A.2 – Dados do 7º problema [46]: Matriz de incidência máquina-peça (24 peças e 14 máquinas).

APÊNDICE B – MINIMIZAÇÃO DA VARIAÇÃO DA CARGA DE TRABALHO NAS CÉLULAS USANDO UM MODELO ESTENDIDO

Nós estendemos o modelo Z_2 para incluir rotas alternativas para as peças e permitir explicitamente a contabilização de todas as operações não-consecutivas da peça j para o cálculo da carga de trabalho induzida por esta peça na máquina i. O modelo Z_2 estendido é dado a seguir:

$$Z_2 = \sum_{i=1}^{M} \sum_{s=1}^{L} X_{is} \cdot \sum_{j=1}^{N} \left(w_{ij} - m_{sj} \right) \tag{B.1}$$

Portanto, nós desejamos minimizar Z_2 sujeito a

$$\sum_{p=1}^{P_j} Y_{jp} = 1 \ (\forall j) \tag{B.2}$$

$$\sum_{s=1}^{L} X_{is} = 1 \ (\forall i) \tag{B.3}$$

$$\sum_{i=1}^{M} X_{is} \geq 2 \ (\forall s) \tag{B.4}$$

$$X_{is} = \{0,1\} \ \ (\forall i,s) \tag{B.5}$$

onde

$$w_{ij} = N_j \cdot \sum_{p=1}^{P_j} Y_{jp} \cdot \left[\left(\sum_{k=1}^{n_{jip}} t_{ijkp} \right) \Big/ T_i \right], \tag{B.6}$$

$$m_{sj} = \left(\sum_{i=1}^{M} X_{is} \cdot w_{ij} \Big/ \sum_{i=1}^{M} X_{is} \right). \qquad \text{(B.7)}$$

B.1 UMA ILUSTRAÇÃO DO MODELO Z_2

Com o objetivo de ilustrar o modelo Z_2, considere um hipotético sistema de manufatura com 3 máquinas e 3 peças. A capacidade das máquinas é mostrada na Tabela B.1 e as informações a respeito das peças é mostrado na tabela B.2. Considere ainda que a configuração das rotas escolhidas é [1 1 1] e que a matriz máquina-célula X é mostrada na Figura B.1.

Tabela B.1 – Capacidade das máquinas (hora/ano)

Máquinas	1	2	3
Capacidade	1820	1600	1800

Tabela B.2 – Informações sobre as peças: demanda, rotas, máquinas.

Peças	**Demanda**	**Rotas**	**Máquinas***
1	265	1	1(2), 2(2), 1(1)
		2	1(4), 3(1)
2	232	1	3(4), 1(2)
		2	2(2), 3(1), 1(2)
3	160	1	2(3)
		2	2(1), 1(2)

* Os valores dentro dos parênteses indicam o tempo de processamento da peça j na máquina i.

$$[X_{is}] = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figura B.1 – Matriz máquina-célula: cada elemento não nulo significa que a máquina i pertence a célula s.

Utilizando a equação (B.6), nós podemos calcular agora a carga de trabalho em cada máquina:

$$w_{11} = N_1 \cdot \left\{ Y_{11} \cdot \left[(t_{1111} + t_{1121}) / T_1 \right] + Y_{11} \cdot \left(t_{1112} / T_1 \right) \right\} = 265 \cdot \left\{ 1 \cdot \left[(2+1) / 1820 \right] + 0 \cdot (4 / 1820) \right\} = 0.44;$$

$$w_{12} = N_2 \cdot \left\{ Y_{21} \cdot \left(t_{1211} / T_1 \right) + Y_{22} \cdot \left(t_{1212} / T_1 \right) \right\} = 232 \cdot \left\{ 1 \cdot (2 / 1820) + 0 \cdot (2 / 1820) \right\} = 0.25;$$

$$w_{13} = N_3 \cdot \left\{ Y_{32} \cdot \left(t_{1312} / T_1 \right) \right\} = 0;$$

$$w_{21} = N_1 \cdot \left\{ Y_{11} \cdot \left(t_{2111} / T_2 \right) \right\} = 290 \cdot \left\{ 1 \cdot (2 / 1600) \right\} = 0.36;$$

$$w_{22} = N_2 \cdot \left\{ Y_{22} \cdot \left(t_{2212} / T_2 \right) \right\} = 0;$$

$$w_{23} = N_3 \cdot \left\{ Y_{31} \cdot \left(t_{2311} / T_2 \right) + Y_{32} \cdot \left(t_{2312} / T_1 \right) \right\} = 160 \cdot \left\{ 1 \cdot (3 / 1600) + 0 \cdot (2 / 1600) \right\} = 0.3;$$

$$w_{31} = N_1 \cdot \left\{ Y_{12} \cdot \left(t_{3112} / T_3 \right) \right\} = 0;$$

$$w_{32} = N_2 \cdot \left\{ Y_{21} \cdot \left({}^{t_{3211}}\!/_{T_2} \right) + Y_{22} \cdot \left({}^{t_{2312}}\!/_{T_1} \right) \right\} = 232 \cdot \left\{ 1 \cdot \left({}^{5}\!/_{1800} \right) + 0 \cdot \left({}^{1}\!/_{1800} \right) \right\} = 0.64;$$

$w_{33} = 0.$

A matriz de incidência máquina-peça $[w_{ij}]_{M*N}$ é mostrada na Figura B.2.

$$[w_{ij}] = \begin{bmatrix} 0.44 & 0.36 & 0.00 \\ 0.25 & 0.00 & 0.64 \\ 0.00 & 0.30 & 0.00 \end{bmatrix}$$

Figura B.2 – Matriz de incidência máquina-peça: cada elemento não nulo especifica a carga de trabalho induzida

O cálculo da carga de trabalho média de cada célula pode ser agora calculado:

$$m_{11} = {}^{(X_{11} \cdot w_{11} + X_{21} \cdot w_{21} + X_{31} \cdot w_{31})}\!/_{X_{11} + X_{21} + X_{31}} =$$

$${}^{(1 \cdot 0.44 + 1 \cdot 0.25 + 0 \cdot 0)}\!/_{2} = 0.35$$

$$m_{12} = {}^{(X_{11} \cdot w_{12} + X_{21} \cdot w_{22} + X_{31} \cdot w_{32})}\!/_{X_{11} + X_{21} + X_{31}} =$$

$$m_{12} = {}^{(1 \cdot 0.36 + 1 \cdot 0 + 0 \cdot 0.30)}\!/_{2} = 0.18$$

$$m_{13} = {}^{(X_{11} \cdot w_{13} + X_{21} \cdot w_{23} + X_{31} \cdot w_{33})}\!/_{X_{11} + X_{21} + X_{31}} =$$

$${}^{(1 \cdot 0 + 1 \cdot 0.64 + 0 \cdot 0)}\!/_{2} = 0.32$$

$$m_{21} = {}^{(X_{12} \cdot w_{11} + X_{22} \cdot w_{21} + X_{32} \cdot w_{31})}\!/_{X_{12} + X_{22} + X_{32}} =$$

$${}^{(0 \cdot 0.44 + 0 \cdot 0.25 + 1 \cdot 0)}\!/_{1} = 0$$

$$m_{22} = {}^{(X_{12} \cdot w_{12} + X_{22} \cdot w_{22} + X_{32} \cdot w_{32})}\!/_{X_{12} + X_{22} + X_{32}} =$$

$${}^{(0 \cdot 0.36 + 0 \cdot 0 + 1 \cdot 0.30)}\!/_{1} = 0.30$$

A matriz $[m_{sj}]_{L*N}$ é mostrada na Figura B.3.

$$[m_{sj}] = \begin{bmatrix} 0.35 & 0.18 & 0.32 \\ 0.00 & 0.30 & 0.00 \end{bmatrix}$$

Figura B.3 – Matriz de de carga média: cada elemento não nulo especifica a carga média induzida pela peça j na célula s.

Desenvolvendo a equação (B.1) nós obtemos a seguinte expressão:

$$Z_2 = X_{11} \cdot \sum_{j=1}^{3} (w_{1j} - m_{1j})^2 + X_{12} \cdot \sum_{j=1}^{3} (w_{1j} - m_{2j})^2 + X_{21} \cdot \sum_{j=1}^{3} (w_{2j} - m_{1j})^2 +$$

$$X_{22} \cdot \sum_{j=1}^{3} (w_{2j} - m_{2j})^2 + X_{31} \cdot \sum_{j=1}^{3} (w_{3j} - m_{1j})^2 + X_{11} \cdot \sum_{j=1}^{3} (w_{3j} - m_{2j})^2 = 0.29$$

Entretanto, se a matriz máquina-peça X considerada for aquela mostrada na Figura B.4, então o valor do modelo Z_2 será igual a 0.0986. Note que uma nova matriz de carga média $[m]_{sj}$ deve ser calculada. Segundo o modelo Z_2 eStendido, o agrupamento de máquinas da Figura B.4 é melhor do que o agrupamento da Figura B.1.

$$[X_{is}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figura B.4– Uma outra configuração da matriz máquina-célula.

APÊNDICE C – CODIFICAÇÃO DOS INDIVÍDUOS

As formas de codificação adotadas neste trabalho são exemplificadas nas Figuras C.1 e C.2 (os indivíduos mostrados codificam 9 tipos de máquinas e 5 tipos de peças – o número de células é 4 e todas os tipos de peças possuem 2 rotas de processamento).

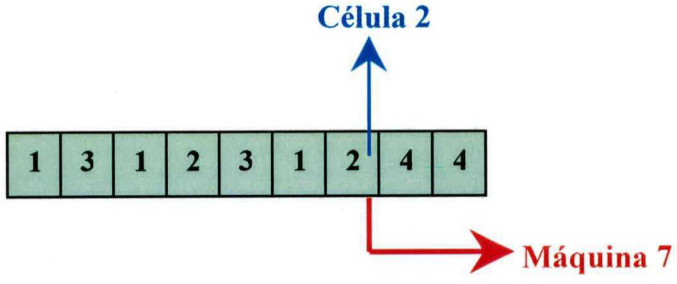


Figura C.1 – Representação simples de um indivíduo (somente máquinas são codificadas).

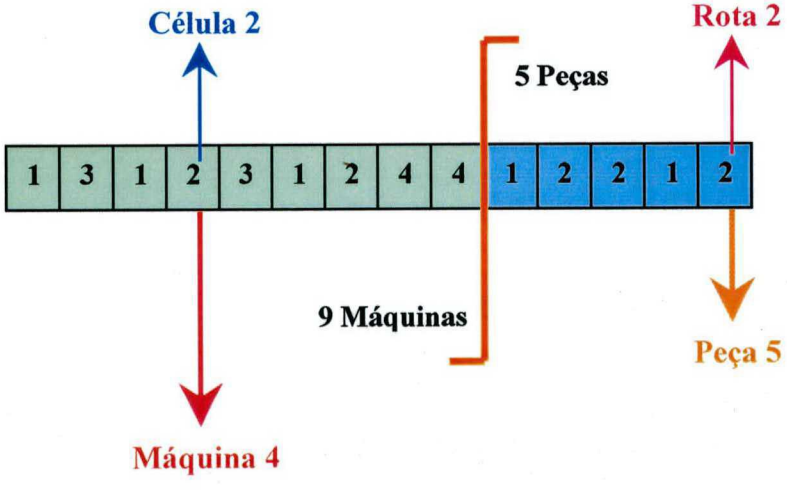


Figura C.2 – Representação composta de um indivíduo (máquinas e peças são codificadas juntas).

APÊNDICE D – RECOMBINAÇÃO E MUTAÇÃO

O processo de recombinação de dois indivíduos, que codificam 9 tipos de máquinas (o número máximo de células é 4, sendo que ele é definido *a priori*) e 5 tipos de peças (exceto o primeiro tipo, todas as outras possuem 2 rotas de processamento), é apresentado na Figura D.1.

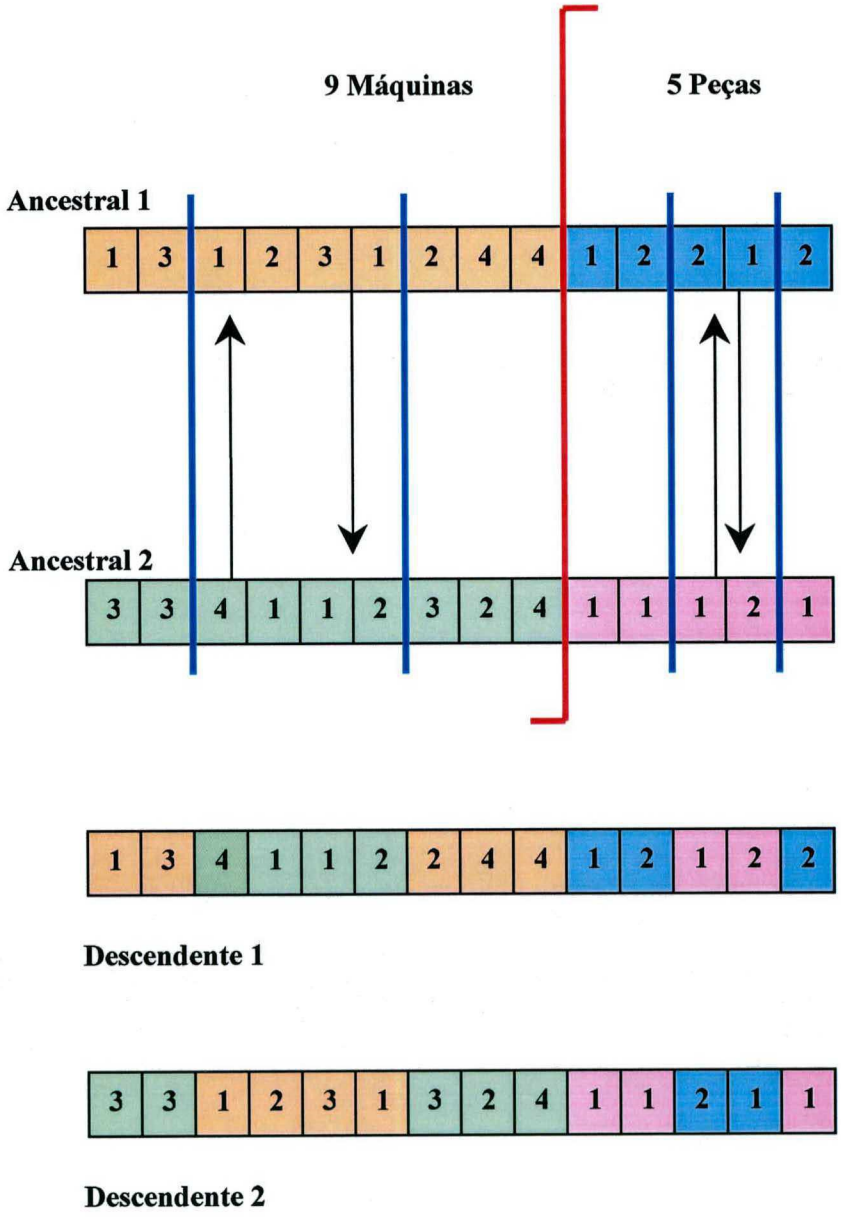


Figura D.1 – Recombinação multiponto entre dois indivíduos que codificam 9 tipos de máquinas e 5 tipos de peças.

A Figura D.1 apresenta 2 pontos de corte para cada parte do indivíduo (máquinas e peças), entretanto, o operador de recombinação multiponto utilizado neste trabalho pode ter um número de pontos de corte maior que 2 (para cada parte). Para os problemas que não envolvem rotas alternativas (i.e., não possuem a parte referente aos tipos de peças), somente 2 pontos de corte são utilizados para cada indivíduo.

O processo de mutação utilizado neste trabalho é exemplificado na Figura D.2. Novamente, o número de células é 4 e o primeiro tipo de peça possui somente 1 rota de processamento (por causa disto, o valor contido nesta posição não pode ser alterada pelo operador de mutação). Para os tipos de máquinas codificados, cada posição que sofre mutação está livre para assumir um valor aleatório dentro do intervalo [1, 4], i.e., de 1 até o número máximo de células (definido a *priori*). O intervalo de valores para cada tipo de peça depende do número de rotas alternativas que cada uma possui.

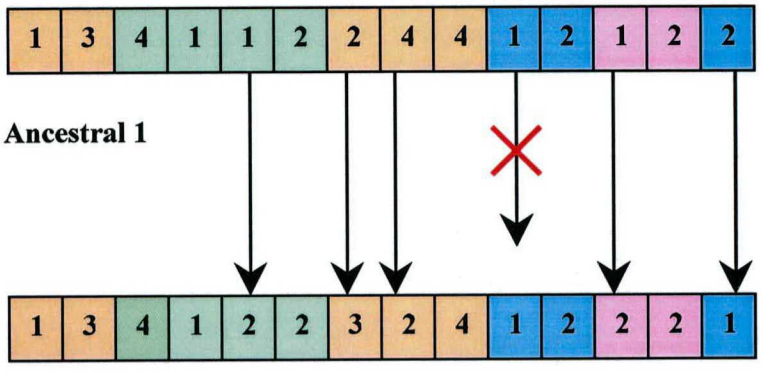


Figura D.2 – Mutação em um indivíduo que codifica 9 tipos de máquinas e 5 tipos de peças.

APÊNDICE E – ALGORITMO HÍBRIDO AG/TS

O algoritmo híbrido AG/TS utilizado neste trabalho é apresentado na Figura E.1. O algoritmo genético sem busca local não possui os blocos mostrados dentro da área pontilhada. Neste caso, o fluxo de execução segue direto do bloco **mutar** para a **2º condição de teste** (se o número de gerações corrente é igual ao número máximo de gerações definido pelo usuário). A **1º condição de teste** verifica se o módulo entre o número de gerações corrente e o parâmetro TBL (tempo de busca local) é igual a zero. Se for, a busca local usando TS é realizada.

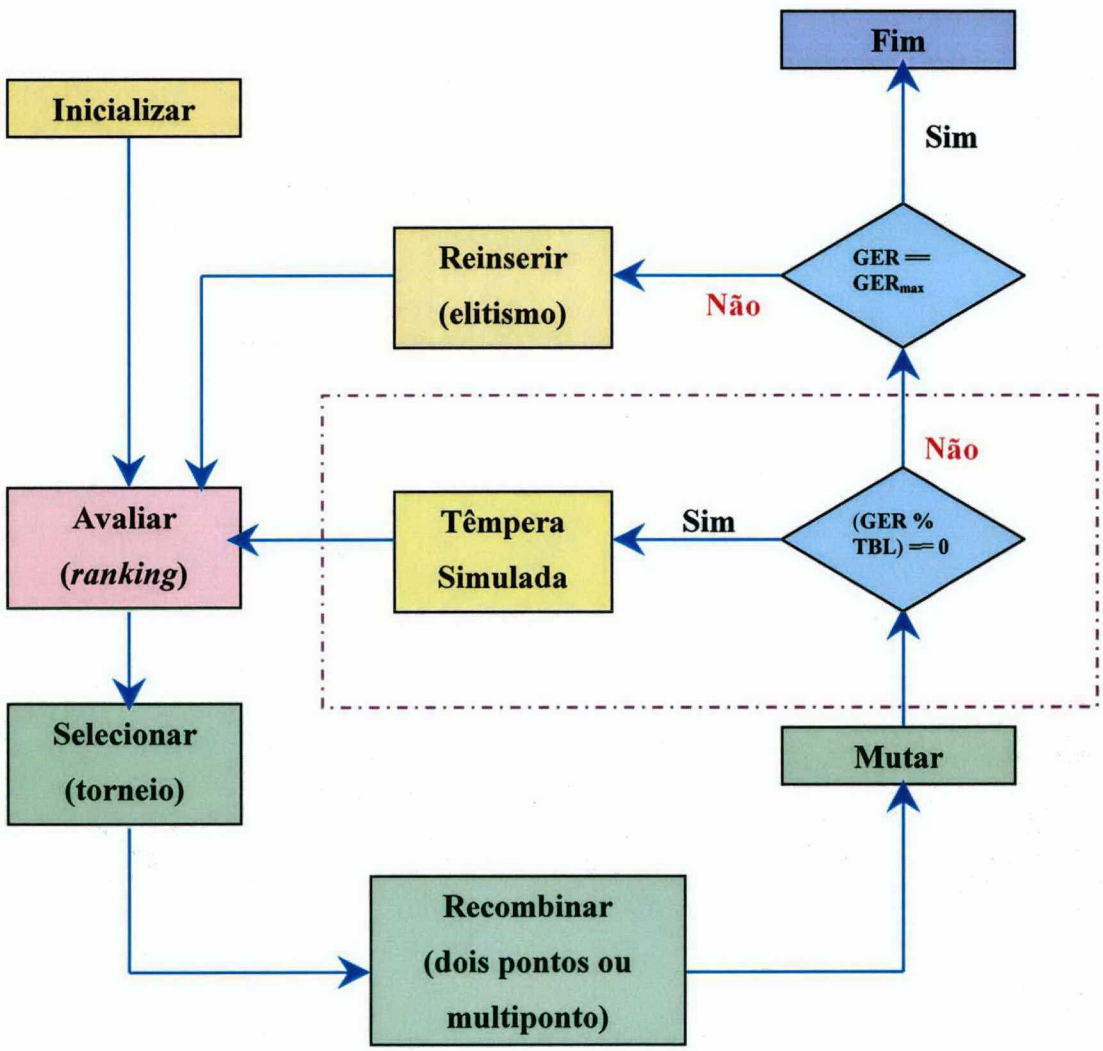


Figura E.1 – Diagrama de Fluxo do algoritmo híbrido AG/TS.